Discrete Applied Mathematics 211 (2016) 183–203

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

The assignment problem with nearly Monge arrays and incompatible partner indices

C. Weiß [a], S. Knust [b], N.V. Shakhlevich [a,*], S. Waldherr [b]

[a] School of Computing, University of Leeds, Leeds LS2 9JT, UK

[b] University of Osnabrück, Institute of Computer Science, 49069 Osnabrück, Germany

ARTICLE INFO

Article history:
Received 27 April 2015
Received in revised form 8 March 2016
Accepted 18 April 2016
Available online 1 June 2016

Keywords:
Assignment problem
Monge property
Monge array

ABSTRACT

In this paper we study the d-dimensional assignment problem in which entries of the cost array satisfy the Monge property, except for ∞-entries, which may violate it. We assume that the ∞-entries are incurred by incompatible partner indices and their number is bounded by an upper bound λ for each index. We show that the problem can be solved in linear time for fixed d and λ, and it becomes strongly NP-hard if d or λ is part of the input.

1. Introduction

The assignment problem is an important combinatorial optimization problem with many applications in different areas (see, e.g., [4]). In this problem the objective is to assign n items from one set $I = \{1, \ldots, n\}$ to n items of another set $J = \{1, \ldots, n\}$, where I may represent a set of workers, jobs, transmitting devices, and J may correspond to machines, rooms, receivers, etc. In the linear assignment problem, additionally an $n \times n$ weight (cost) matrix $W = (w_{ij})$ is given, and the goal is to find an assignment with minimum total weight.

An *assignment* S may be represented by a set of n pairs, $S = \{(i^1, j^1), \ldots, (i^n, j^n)\}$ with $\{i^1, i^2, \ldots, i^n\} = \{j^1, j^2, \ldots, j^n\} = \{1, 2, \ldots, n\}$ and it is characterized by the weight

$$w(S) = \sum_{(i,j)\in S} w_{ij}.$$

An assignment S is also called a *solution* and a solution S^* with minimum weight is called an *optimal solution*.

* Corresponding author.
E-mail addresses: mm12cw@leeds.ac.uk (C. Weiß), sknust@uni-osnabrueck.de (S. Knust), N.Shakhlevich@leeds.ac.uk (N.V. Shakhlevich), swaldher@uni-osnabrueck.de (S. Waldherr).

http://dx.doi.org/10.1016/j.dam.2016.04.019

An alternative representation of the assignment problem uses binary decision variables x_{ij} indicating the assignment of i-items to j-items:

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} = 1, \qquad 1 \le i \le n, \\
& \sum_{i=1}^{n} x_{ij} = 1, \qquad 1 \le j \le n, \\
& x_{ij} \in \{0,1\}, \qquad 1 \le i,j \le n.
\end{aligned}$$

Clearly, there exists a one-to-one correspondence between any solution S and the binary solution matrix $X_S = (x_{ij})$ with $x_{i^k j^k} = 1$ for every pair $(i^k, j^k) \in S$, $1 \le k \le n$. The linear assignment problem can be solved in polynomial time, e.g., by the Hungarian algorithm [21], which can be implemented to run in $O(n^3)$ time [4].

The extension of the linear assignment to the multi-dimensional case gives rise to the so called *axial d-dimensional assignment problem* ($d \ge 2$): given a d-dimensional $n \times \cdots \times n$ weight array $W = (w_{i_1 \ldots i_d})$, an assignment S is a set of n d-tuples $\{(i_1^1, \ldots, i_d^1), (i_1^2, \ldots, i_d^2), \ldots, (i_1^n, \ldots, i_d^n)\}$ with $\{i_\ell^1, \ldots, i_\ell^n\} = \{1, \ldots, n\}$ for all $\ell = 1, \ldots, d$, and its weight is

$$w(S) = \sum_{(i_1,\ldots,i_d)\in S} w_{i_1 \ldots i_d}.$$

The formulation in terms of the binary decision variables $x_{i_1 \ldots i_d}$ is as follows:

$$\begin{aligned}
\min \quad & \sum_{i_1,\ldots,i_d} w_{i_1 \ldots i_d} x_{i_1 \ldots i_d} \\
\text{s.t.} \quad & \sum_{\substack{i_1,\ldots,i_d \\ \text{s.t. } i_\ell = k}} x_{i_1 \ldots i_d} = 1, \quad 1 \le \ell \le d,\ 1 \le k \le n, \\
& x_{i_1 i_2 \ldots i_d} \in \{0,1\}, \qquad 1 \le i_1, i_2, \ldots, i_d \le n.
\end{aligned} \tag{1}$$

The associated solution array $X_S = (x_{i_1 \ldots i_d})$ has a 1-entry $x_{i_1 \ldots i_d} = 1$ for every d-tuple $(i_1, \ldots, i_d) = (i_1^k, \ldots, i_d^k) \in S$, where $1 \le k \le n$. Note that in any solution X_S, the number of 1-entries is n, while the remaining $n^d - n$ entries are 0. Unlike the two-dimensional version, the d-dimensional assignment problem is strongly NP-hard for each fixed $d \ge 3$, see, e.g., [19].

The assignment problem is strongly related to the transportation problem. In its usual, continuous form, the d-dimensional transportation problem is formulated as follows:

$$\begin{aligned}
\min \quad & \sum_{i_1,\ldots,i_d} w_{i_1 \ldots i_d} x_{i_1 \ldots i_d} \\
\text{s.t.} \quad & \sum_{\substack{i_1,\ldots,i_d \\ \text{s.t. } i_\ell = k}} x_{i_1 \ldots i_d} = a_k^\ell, \quad 1 \le \ell \le d,\ 1 \le k \le n, \\
& x_{i_1 i_2 \ldots i_d} \ge 0, \qquad 1 \le i_1, i_2, \ldots, i_d \le n.
\end{aligned}$$

Here a_k^ℓ are given non-negative supply/demand-values satisfying $\sum_{k=1}^n a_k^1 = \sum_{k=1}^n a_k^2 = \cdots = \sum_{k=1}^n a_k^d$. The assignment problem is the special case of the integer version of the transportation problem, with $a_k^\ell = 1$ for all $k = 1, \ldots, n$ and $\ell = 1, \ldots, d$.

It is well known that many combinatorial optimization problems, including the assignment problem and the transportation problem, can be solved faster (by a greedy algorithm) if the weight matrix is a Monge matrix. We remind here the main definitions and results following the survey paper [5]. An $n \times n$ matrix $W = (w_{ij})$ is called a *Monge matrix* if for all row indices $1 \le i < r \le n$ and all column indices $1 \le j < s \le n$ the so-called Monge property is satisfied:

$$w_{ij} + w_{rs} \le w_{is} + w_{rj}. \tag{2}$$

An optimal solution to the assignment problem with a Monge matrix is given by the n pairs $(1, 1), (2, 2), \ldots, (n, n)$.

More generally, an $n \times \cdots \times n$ array $W = (w_{i_1 \ldots i_d})$ is called a (d-dimensional) Monge array if for all $i_\ell, j_\ell \in \{1, \ldots, n\}$, $\ell = 1, \ldots, d$, we have

$$w_{s_1 s_2 \ldots s_d} + w_{t_1 t_2 \ldots t_d} \le w_{i_1 i_2 \ldots i_d} + w_{j_1 j_2 \ldots j_d}, \tag{3}$$

where $s_\ell = \min\{i_\ell, j_\ell\}$ and $t_\ell = \max\{i_\ell, j_\ell\}$ for $\ell = 1, \ldots, d$. An optimal solution to the multi-dimensional assignment problem with a Monge array is given by the n d-tuples $(1, \ldots, 1), (2, \ldots, 2), \ldots, (n, \ldots, n)$.

A number of typical scenarios with Monge and Monge-like arrays are discussed in the survey papers [3] and [5]. Several practical applications give rise to ∞-entries in the weight array W. The purpose of infinity entries is to model forbidden

assignments, i.e., if $w_{ij} = \infty$, then i cannot be assigned to j. In this situation, $a + \infty = \infty$ for all $a \in \mathbb{R} \cup \{\infty\}$ and $a < \infty$ for all $a \in \mathbb{R}$. Depending on the position of the ∞-entries, the Monge property may still be satisfied, so that a greedy solution to the assignment and transportation problem remains optimal. For example, if in a Monge matrix all entries in the lower triangle are replaced by ∞ ($w_{ij} = \infty$ for all $i > j$) or if all entries in the upper triangle are replaced by ∞ ($w_{ij} = \infty$ for all $i < j$), then the resulting matrix still satisfies the Monge property.

In the multi-dimensional case, transportation and assignment problems remain greedily solvable in the presence of forbidden entries, if the incurred ∞-entries do not destroy the Monge property. This is shown, e.g. in [26], which exploits the relationship between multi-dimensional Monge arrays (with and without infinities) and submodular functions. Note that the requirement of [26] that the finite entries form a sublattice implies that ∞-entries do respect the Monge property.

In general, however, an arbitrary introduction of ∞-entries may destroy the Monge property in a matrix that initially satisfied it. In [9] so-called incomplete Monge matrices are studied where some values in the matrix are not specified and the Monge property must only hold for any four specified entries w_{ij}, w_{rs}, w_{is}, w_{rj}. In the following we introduce a related concept where unspecified entries are replaced by infinity, which implies that these assignments are forbidden. We call a weight matrix W *nearly Monge* if all quadruples of finite entries w_{ij}, w_{rs}, w_{is}, w_{rj} satisfy the Monge property (2), while quadruples with ∞-entries may violate it. Similarly, in the multi-dimensional case, a d-dimensional array W is called nearly Monge if condition (3) is satisfied for all finite entries.

In this paper, we study an important subclass of such matrices and arrays where ∞-entries are incurred by incompatible partner indices. In the two-dimensional case, if two indices $i = i^*, j = j^*$ are incompatible, we call them *incompatible partners*. Then the corresponding cost is $w_{i^*j^*} = \infty$ to make the assignment (i^*, j^*) *forbidden*. We assume that a parameter $\lambda \leq n$ is given which denotes an upper bound on the number of incompatible partners for every row index $i = i^*$ and every column index $j = j^*$. This implies that there are at most λ forbidden entries in each row and in each column of the matrix W.

In the multi-dimensional assignment problem, denoted by AP(d, λ), each pair of incompatible partners $i_u = i_u^*, i_v = i_v^*$ incurs forbidden entries for all d-tuples of the form $\left(i_1, \ldots, i_u^*, \ldots, i_v^*, \ldots, i_d\right)$, where all indices, except i_u and i_v, take all possible values from $\{1, \ldots, n\}$. Each d-tuple that contains at least one pair of incompatible partners is forbidden; the corresponding w-value is ∞. Again we assume that we have an upper bound λ, which limits for each index $i_u = i_u^*$ the number of incompatible partners for each $v \neq u$. This means that for each value i_u^* and each $v \neq u$ at most λ values $i_{v,1}^*, i_{v,2}^*, \ldots, i_{v,\lambda}^*$ exist such that all assignments $(i_1, \ldots, i_{u-1}, i_u = i_u^*, i_{u+1}, \ldots, i_{v-1}, i_v = i_{v,\mu}^*, i_{v+1}, \ldots, i_d)$ are forbidden for $\mu = 1, 2, \ldots, \lambda$. Hence, in this case the total number of incompatible partners for $i_u = i_u^*$ is at most

$$\Omega = \lambda(d - 1). \tag{4}$$

Example 1. As an example of problem AP(3, 1) consider the nearly Monge array W with $n = 3$ where all entries are equal to 1 except for the ∞-entries. If the first pair of incompatible partners is $i_1^* = 2, i_2^* = 3$, then all assignments $(2, 3, i_3)$ with $i_3 \in \{1, 2, 3\}$ are forbidden, i.e. the ∞-entries are $w_{2,3,1} = w_{2,3,2} = w_{2,3,3} = \infty$. The incompatible pair $i_1^* = 1, i_3^* = 3$ leads to the ∞-entries $w_{1,1,3} = w_{1,2,3} = w_{1,3,3} = \infty$. The incompatible pair $i_2^* = 3, i_3^* = 3$ incurs $w_{1,3,3} = w_{2,3,3} = w_{3,3,3} = \infty$. This situation corresponds to $\lambda = 1$.

An example of problem AP(3, 2) with $\lambda = 2$ can be obtained from the previous example of AP(3, 1) by adding more incompatible pairs. If, for example, the pair $i_1^* = 1, i_3^* = 2$ is added, the index $i_1^* = 1$ gets a second incompatible partner for $v = 3$. Note that if instead we add the incompatible pair $i_1^* = 1, i_2^* = 2$, then $\lambda = 1$ remains unchanged, as the index $i_1^* = 1$ did not have an incompatible partner for $v = 2$ before.

Applications of assignment problems with incompatible partners are typical for scenarios where items from different sets have to be matched, but there is a certain number of combinations that are forbidden. For example, in timetabling the allocation of certain classes to some classrooms or time slots has to be avoided. In scheduling problems with unit processing times and arbitrary release dates and deadlines (which can be modeled as an assignment problem), it is not allowed to allocate a job to a time slot before its release date or after its deadline. In scheduling problems with multiple machines there can be additional constraints that do not allow some job–machine pairs. In transportation scheduling, allocations of certain types of vehicles to some routes can be forbidden; in addition there may be restrictions on drivers' allocation.

Assignment problems with nearly Monge arrays defined by incompatible partners arise for example in applications related to satellite communication or in synchronous open shop scheduling. There also exists a close relation to the well-known max-weight edge coloring problem on bipartite graphs. We discuss the two applications in more detail in Section 2 and the relation to max-weight edge coloring in Section 3.

The remainder of the paper is organized as follows. In Section 2 we present two applied scenarios that can be modeled as problem AP(d, 1) with nearly Monge arrays. In Section 3 we show that problem AP(d, λ) is strongly NP-hard if one of the parameters, d or λ, is part of the input. Following that, in Section 4 we present some basic properties of nearly Monge matrices. In Sections 5 and 6 we study the problem with fixed d and λ, which is a typical assumption in applications. In Section 5 we formulate a "corridor property" that characterizes the structure of an optimal solution. It implies that 1-entries of an optimal solution array belong to a corridor of limited width around the main diagonal. Based on that property, in Section 6 we present an efficient algorithm that solves problem AP(d, λ) in linear time for fixed d and λ, and in FPT time for the parameters d and λ. This has new implications for the complexity of the applications in satellite communications,

synchronous open shop scheduling and max-weight edge coloring. In Section 7 the corridor property for other versions of the assignment problem is discussed. Finally, conclusions are presented in Section 8.

2. Applications

In this section we describe two applications related to satellite communication and synchronous open shop scheduling that can be modeled as problem AP(d, λ) with a d-dimensional nearly Monge weight array and $\lambda = 1$.

In both applications, there are given d n-tuples $\Gamma^\ell = \left(\gamma_1^\ell, \gamma_2^\ell, \ldots, \gamma_n^\ell\right)$, $1 \le \ell \le d$, and the associated assignment problems have a cost array W of the form

$$w_{i_1 \ldots i_d} = \max\left\{\gamma_{i_1}^1, \gamma_{i_2}^2, \ldots, \gamma_{i_d}^d\right\}. \tag{5}$$

If the numbers in each n-tuple Γ^ℓ are listed in non-decreasing order, then W is a Monge array [2]. This can be seen as follows. If we choose $w_{i_1 i_2 \ldots i_d}$, $w_{j_1 j_2 \ldots j_d}$, $w_{s_1 s_2 \ldots s_d}$ and $w_{t_1 t_2 \ldots t_d}$ as in the definition of the multi-dimensional Monge property (3), then due to the maximum-definition and the ordering at least one of the entries $w_{i_1 i_2 \ldots i_d}$ or $w_{j_1 j_2 \ldots j_d}$ is at least as large as $w_{t_1 t_2 \ldots t_d}$. Furthermore, again due to the maximum-definition and the ordering, each entry $w_{i_1 i_2 \ldots i_d}$ and $w_{j_1 j_2 \ldots j_d}$ is at least as large as $w_{s_1 s_2 \ldots s_d}$.

The assignment problem that arises in the context of satellite communication has been under study since the 80s, see, e.g., [15,20,27,28]. In this problem a bipartite graph $G = (V_1 \cup V_2, E)$ is given where the vertex sets V_1 and V_2 correspond to senders and receivers and the edges E model transmissions. We assume $|V_1| = d$, $|V_2| = n$ with $d \le n$. For any transmission $(\ell, i) \in E$ from sender $\ell \in V_1$ to receiver $i \in V_2$, there is given a transmission time $t_{\ell i}$. Each sender can send at most one message at any time, and each receiver can receive at most one message at any time, while independent senders/receivers can perform transmissions in parallel. Switches in traffic are made simultaneously, so that a feasible solution can be characterized by a set of periods, each of which does not involve the same sender or the same receiver more than once. Hence, each period is described by at most d pairs (ℓ, i) denoting messages from senders $\ell \in V_1$ to receivers $i \in V_2$ that can be sent simultaneously. A feasible solution consisting of κ periods is a partition $E_1 \cup E_2 \cup \cdots \cup E_\kappa$ of the edge set E such that no two edges of the same set E_q are incident to the same vertex. The duration of a period $q \in \{1, \ldots, \kappa\}$ corresponds to the longest transmission of that period, i.e.

$$w(E_q) = \max\{t_{\ell i} \mid (\ell, i) \in E_q\},$$

and the total transmission time is equal to $\sum_{q=1}^{\kappa} w(E_q)$.

We now model this problem as an assignment problem. Each period can be described by d pairs $(1, i_1), \ldots, (d, i_d)$ denoting that for $\ell = 1, \ldots, d$, messages are sent from $\ell \in V_1$ to $i_\ell \in V_2$. For simplicity we assume here that each sender sends a message in every time period; the general case, where senders are allowed to be idle in some periods, will be discussed at the end of Section 3. The entries of the cost array W correspond to the durations of the periods, i.e. for any selection of d receivers $i_1, i_2, \ldots, i_d \in V_2$ we define

$$w_{i_1 \ldots i_d} = \begin{cases} \max\left\{t_{1,i_1}, t_{2,i_2}, \ldots, t_{d,i_d}\right\}, & \text{if all receivers } i_1, i_2, \ldots, i_d \text{ are different,} \\ \infty, & \text{otherwise.} \end{cases}$$

Note that $w_{i_1 \ldots i_d} = \infty$ if and only if $i_j = i_k$ for some $1 \le j \ne k \le d$, i.e., if the same receiver is activated simultaneously by two different senders j and k in one period. Thus, each index in the W-matrix has exactly one incompatible partner in each other dimension, which means that $\lambda = 1$ is an upper bound on the number of incompatible partner indices.

In order to achieve the Monge property for finite entries of the array $W = (w_{i_1 \ldots i_d})$, we define for each sender ℓ the n-tuple $\Gamma^\ell = \left(\gamma_1^\ell, \gamma_2^\ell, \ldots, \gamma_n^\ell\right)$ by the durations $t_{\ell i}$ of the messages to be sent from ℓ to all receivers $i \in V_2$, and renumber the γ-values so that

$$\gamma_1^\ell \le \gamma_2^\ell \le \cdots \le \gamma_n^\ell. \tag{6}$$

Then the array $\hat{W}$ of weights

$$\hat{w}_{i_1 \ldots i_d} = \begin{cases} \max\left\{\gamma_{i_1}^1, \gamma_{i_2}^2, \ldots, \gamma_{i_d}^d\right\}, & \text{if all messages corresponding to } \gamma_{i_1}^1, \ldots, \gamma_{i_d}^d \text{ have different receivers,} \\ \infty, & \text{otherwise,} \end{cases} \tag{7}$$

is a permutation of W of type (5), apart from the infinities, and therefore all finite entries satisfy the Monge property.

Example 2. Consider an example with $d = 2$ senders, $n = 4$ receivers, and the following transmission times $t_{\ell i}$ for senders $\ell \in V_1 = \{1, 2\}$ and receivers $i \in V_2 = \{1, 2, 3, 4\}$:

$\ell \setminus i$	1	2	3	4
1	5	2	7	3
2	4	2	3	6

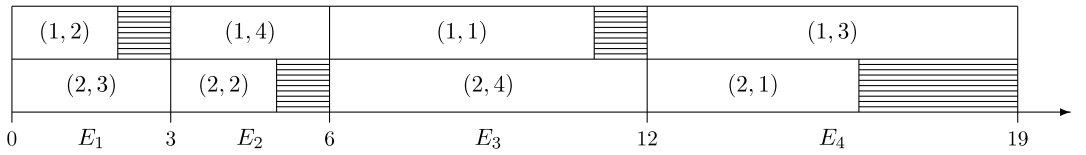

Fig. 1. A feasible transmission schedule for Example 2.

Then the associated matrix W is of the form

$$W = \left(\begin{array}{c|cccc} & (2,1) & (2,2) & (2,3) & (2,4) \\ \hline (1,1) & \infty & 5 & 5 & 6 \\ (1,2) & 4 & \infty & 3 & 6 \\ (1,3) & 7 & 7 & \infty & 7 \\ (1,4) & 4 & 3 & 3 & \infty \end{array}\right),$$

where pairs (ℓ, i) denote the messages. Notice that matrix W is not nearly Monge.

Consider matrix $\hat{W}$ obtained from W using $\Gamma^1 = (t_{12}, t_{14}, t_{11}, t_{13}) = (2, 3, 5, 7)$ and $\Gamma^2 = (t_{22}, t_{23}, t_{21}, t_{24}) = (2, 3, 4, 6)$:

$$\hat{W} = \left(\begin{array}{c|cccc} & (2,2) & (2,3) & (2,1) & (2,4) \\ \hline (1,2) & \infty & 3 & 4 & 6 \\ (1,4) & 3 & 3 & 4 & \infty \\ (1,1) & 5 & 5 & \infty & 6 \\ (1,3) & 7 & \infty & 7 & 7 \end{array}\right).$$

Matrix $\hat{W}$ is nearly Monge, but not Monge.

A feasible solution is given by $E_1 = \{(1, 2), (2, 3)\}$, $E_2 = \{(1, 4), (2, 2)\}$, $E_3 = \{(1, 1), (2, 4)\}$ and $E_4 = \{(1, 3), (2, 1)\}$. The corresponding schedule is shown in Fig. 1; its total transmission time is $3 + 3 + 6 + 7 = 19$, which can be shown to be optimal.

As described above, $\lambda = 1$ corresponds to the situation that messages to the same receiver are incompatible, i.e. cannot be sent simultaneously. In a more general setting, also messages to receivers in close proximity to each other may be incompatible, e.g., due to interference. If, for example, for each receiver one or two 'neighbors' are affected, we get problems with $\lambda = 2$ or $\lambda = 3$.

As a second application we consider the synchronous open shop scheduling problem, which is a counterpart of the synchronous flow shop problem studied in [31] and [33]. In this problem there are given d machines $M_1, \ldots, M_d$ and n jobs, where job i consists of d operations $O_{1i}, O_{2i}, \ldots, O_{di}$. Operation $O_{\ell i}$ has to be processed without preemption on machine M_ℓ for $p_{\ell i}$ time units. In a feasible schedule each machine processes at most one operation at any time, and each job is processed on at most one machine at any time. The processing is organized in synchronized cycles where jobs have to be moved synchronously to the machines of the next cycle. This means that in a cycle all current jobs start at the same time on the corresponding machines, and those with smaller processing times have to wait until the longest operation of the cycle is finished. Afterwards, all jobs are replaced simultaneously. As a consequence, the processing time of a cycle is determined by the maximum processing time of its operations. The goal is to assign all nd operations to n cycles so that the completion time of the last cycle is minimized.

Any feasible schedule can be characterized by n cycles $(i_1^1, \ldots, i_d^1), (i_1^2, \ldots, i_d^2), \ldots, (i_1^n, \ldots, i_d^n)$ where each cycle $k \in \{1, \ldots, n\}$ is described by d job indices $(i_1^k, \ldots, i_d^k)$, assuming that in cycle k job i_ℓ^k is allocated to machine M_ℓ for $\ell = 1, \ldots, d$.

Similar to the previous problem, the problem of allocating jobs to machines within n cycles can be modeled as an assignment problem with costs $w_{i_1 \ldots i_d}$ for d-tuples $(i_1, \ldots, i_d)$ defined by

$$w_{i_1 \ldots i_d} = \begin{cases} \max\left\{p_{1,i_1}, p_{2,i_2}, \ldots, p_{d,i_d}\right\}, & \text{if all job indices } i_1, i_2, \ldots, i_d \text{ are different,} \\ \infty, & \text{otherwise.} \end{cases}$$

Here ∞-entries prohibit the allocation of two operations of the same job to one cycle. Since for each machine–job pair (ℓ, i_ℓ) exactly one incompatible partner (ℓ', i_ℓ) exists, for each other machine ℓ', $\ell \neq \ell'$, we have $\lambda = 1$.

As in the previous application, in order to achieve the Monge property for finite entries of the array $W = (w_{i_1 \ldots i_d})$, we define for each machine M_ℓ the n-tuple $\Gamma^\ell = \left(\gamma_1^\ell, \gamma_2^\ell, \ldots, \gamma_n^\ell\right)$ by the processing times $p_{\ell i}$ of operations $O_{\ell i}$ $(1 \leq i \leq n)$ that have to be processed on that machine, and renumber the γ-values so that (6) holds. Then the array $\hat{W}$ of weights

$$\hat{w}_{i_1 \ldots i_d} = \begin{cases} \max\left\{\gamma_{i_1}^1, \gamma_{i_2}^2, \ldots, \gamma_{i_d}^d\right\}, & \text{if all job indices corresponding to } \gamma_{i_1}^1, \gamma_{i_2}^2, \ldots, \gamma_{i_d}^d \text{ are different,} \\ \infty, & \text{otherwise,} \end{cases} \tag{8}$$

is of type (5), apart from the infinities, and therefore all finite entries satisfy the Monge property.

Example 2 can be reformulated to define an instance of the synchronous open shop problem, replacing "senders" and "receivers" by "machines"' and "jobs", respectively. The entries of matrices W and $\hat{W}$ remain the same. In the schedule shown in Fig. 1, the sets E_1, E_2, E_3 and E_4 have now the meaning of cycles, while pairs (ℓ, i) represent now operations $O_{\ell i}$.

In addition to the introduced versions of the two applications, in the subsequent sections we will consider their generalized versions as well, where some senders or machines are allowed to stay idle so that less than d activities may occur. Notice that there are instances for which an optimal solution with idle times may outperform any solution without idle times.

3. NP-hardness of problem AP(d, λ)

In this section we establish a link between problem AP(d, λ) and the max-weight edge coloring problem in graphs and use that link to show that problem AP(d, λ) with arbitrary d and fixed λ is strongly NP-hard, even for $\lambda = 1$. Furthermore, we discuss the complementary result that AP(d, λ) with arbitrary λ and fixed d is strongly NP-hard, even if $d = 3$.

Both applications described in Section 2 can be modeled as special cases of the well-known *Max-weight Edge Coloring* problem (MEC). Originated in the area of satellite communication [27], problem MEC has attracted considerable attention of researchers, see e.g., [11,22,23]. The related max-weight vertex coloring problem MVC was studied in [8,10,13]. Since the problem MEC on a graph G is equivalent to coloring the vertices of the line graph $L(G)$, any algorithm for MVC can also be applied to MEC.

In the MEC problem we are given a graph $G = (V, E)$ with vertex set V and edge set E. A feasible edge coloring of G with κ colors is a partition $E_1 \cup E_2 \cup \cdots \cup E_\kappa$ of the edge set E with an assignment of a color c to every subset E_c, $1 \le c \le \kappa$, such that no two edges of the same color c are incident to the same vertex. Additionally, the edges $e \in E$ have weights $w(e)$, and the weight of a feasible edge coloring is defined as $\sum_{c=1}^{\kappa} w(c)$, where $w(c)$ is the maximum weight among the edges that have color c,

$$w(c) = \max\{w(e) | e \in E_c\}. \tag{9}$$

The objective is to find an edge coloring of minimum weight. Note that when introduced in this way, the number of colors is not of importance and we are allowed to use a greater than necessary number of colors if this improves the objective value.

In the following we consider the version of MEC defined on a complete bipartite graph $G = K_{m,n}$ known to be NP-hard in the strong sense [10,27], if both m and n are part of the input. In graph $K_{m,n}$, the vertex set V is partitioned into V_1 and V_2 with $|V_1| = m$, $|V_2| = n$, and the edges E connect every vertex from V_1 with every vertex from V_2. We denote this problem by MEC($K_{m,n}$). It is easy to see that the following observation holds (for satellite communication this has already been noted in, e.g., [10]).

Observation 1. *The problems of scheduling satellite transmissions or synchronous open shops, with idle senders or machines allowed, are equivalent to* MEC($K_{d,n}$) *and therefore they are NP-hard in the strong sense, if both d and n are part of the input.*

We now turn to the main problem AP(d, λ).

Theorem 1. *Problem* MEC($K_{m,n}$) *reduces to problem* AP(d, λ) *with* $d = m$ *and* $\lambda = 1$,

$$\text{MEC}(K_{m,n}) \propto \text{AP}(m, 1). \tag{10}$$

Proof. Without loss of generality we assume that $m \le n$. Clearly, the number of colors κ in any feasible solution to MEC($K_{m,n}$) satisfies $n \le \kappa \le nm$, and the number of edges that get the same color $c \in \{1, \ldots, \kappa\}$ may vary from 1 to m.

We present the proof for $m = 2$ first and then generalize it for an arbitrary m. In order to prove (10) for $m = 2$, we create an auxiliary problem MEC^{2n}($K_{2,2n}$) which corresponds to the MEC for the bipartite graph $K_{2,2n}$ with exactly $2n$ colors allowed. We show that

$$\text{MEC}(K_{2,n}) \propto \text{MEC}^{2n}(K_{2,2n}) \tag{11}$$

and

$$\text{MEC}^{2n}(K_{2,2n}) \propto \text{AP}(2, 1). \tag{12}$$

Given a bipartite graph $G = (V_1 \cup V_2, E)$ for the original problem MEC($K_{2,n}$), create the extended bipartite graph $\widetilde{G} = (V_1 \cup \widetilde{V}_2, \widetilde{E})$ for the auxiliary problem with unchanged first set of vertices V_1, while the second set of vertices V_2 is extended by adding n additional vertices so that $|\widetilde{V}_2| = 2n$. The edge set $\widetilde{E}$ is extended accordingly, to include $mn = 2n$ new edges connecting all vertices from V_1 with the n auxiliary vertices from $\widetilde{V}_2$, the weights of those edges being set to 0. Clearly, any feasible solution to MEC($K_{2,n}$) that uses κ colors can be extended to a feasible solution to MEC^{2n}($K_{2,2n}$) that uses $2n$ colors without changing the objective value such that κ colors contribute to the weight function, while $2n - \kappa$ colors carry 0-weight. Conversely, if we have a solution to MEC^{2n}($K_{2,2n}$), we simply drop all auxiliary edges (having weight 0) and obtain a feasible solution with the same objective value for MEC($K_{2,n}$). Thus, reduction (11) holds.

In what follows we reformulate problem $\text{MEC}^{2n}(K_{2,2n})$ as an assignment problem which finds for every edge $(1, \widetilde{v}_i)$ a mate $(2, \widetilde{v}_j)$ with $\{1, 2\} \subseteq V_1$ and $\{\widetilde{v}_i, \widetilde{v}_j\} \subseteq \widetilde{V}_2$, such that both edges can get the same color, contributing to the objective function the amount $\max\{w(1, \widetilde{v}_i), w(2, \widetilde{v}_j)\}$.

Create two sequences of edges $\left((1, \widetilde{v}_1^1), (1, \widetilde{v}_1^2), \ldots, (1, \widetilde{v}_1^{2n})\right)$ and $\left((2, \widetilde{v}_2^1), (2, \widetilde{v}_2^2), \ldots, (2, \widetilde{v}_2^{2n})\right)$ each consisting of $2n$ edges listed in non-decreasing order of their weights:

$$w(1, \widetilde{v}_1^1) = w(1, \widetilde{v}_1^2) = \cdots = w(1, \widetilde{v}_1^n) \le w(1, \widetilde{v}_1^{n+1}) \le \cdots \le w(1, \widetilde{v}_1^{2n}),$$
$$w(2, \widetilde{v}_2^1) = w(2, \widetilde{v}_2^2) = \cdots = w(2, \widetilde{v}_2^n) \le w(2, \widetilde{v}_2^{n+1}) \le \cdots \le w(2, \widetilde{v}_2^{2n}).$$

Here the first n terms of each list correspond to the edges incident to auxiliary vertices which have 0-weights. We define the matrix W with $2n$ rows and $2n$ columns as follows:

- each row i, $1 \le i \le 2n$, corresponds to the edge $e_1^i = (1, \widetilde{v}_1^i)$;
- each column j, $1 \le j \le 2n$, corresponds to the edge $e_2^j = (2, \widetilde{v}_2^j)$;
- the weight-value w_{ij} corresponds to the weight of assigning the same color to the pair of edges e_1^i and e_2^j,

$$w_{ij} = \begin{cases} \max\{w(e_1^i), w(e_2^j)\}, & \text{if } \widetilde{v}_1^i \ne \widetilde{v}_2^j, \\ \infty, & \text{otherwise.} \end{cases} \tag{13}$$

The matrix with entries $\max\{w(e_1^i), w(e_2^j)\}$ is the two-dimensional case of (5) and therefore satisfies the Monge property. Thus, matrix W defined by (13), where ∞-entries correspond to incompatible pairs of edges e_1^i, e_2^j, is a nearly Monge matrix with $\lambda = 1$ incompatible partner for every $i = i^*$ or $j = j^*$.

It is easy to see that for any feasible solution to problem $\text{MEC}^{2n}(K_{2,2n})$ there exists a feasible solution to problem AP(2, 1) with the same (finite) weight and vice versa. Thus, reduction (12) holds.

Consider now the case of an arbitrary $m \le n$. Given problem $\text{MEC}(K_{m,n})$, introduce the extended problem $\text{MEC}^{mn}(K_{m,mn})$ by adding $(m-1)n$ auxiliary vertices to V_2 and introducing edges of weight 0 that connect them with m vertices from V_1. As before, any feasible solution to $\text{MEC}(K_{m,n})$ with κ colors can be extended to a feasible solution of $\text{MEC}^{mn}(K_{m,mn})$ with mn colors (and vice versa) such that κ colors contribute to the weight function, while $mn - \kappa$ colors carry 0-weight. Thus,

$$\text{MEC}(K_{m,n}) \propto \text{MEC}^{mn}(K_{m,mn}). \tag{14}$$

In order to reduce $\text{MEC}^{mn}(K_{m,mn})$ to AP(m, 1), create m sequences of edges $\left((h, \widetilde{v}_h^1), (h, \widetilde{v}_h^2), \ldots, (h, \widetilde{v}_h^{mn})\right)$ each listed in non-decreasing order of the weights, $1 \le h \le m$. The array W is m-dimensional, with each dimension h corresponding to edges $e_h^i = (h, \widetilde{v}_h^i)$ incident to h, the number of entries in each dimension being mn. The weight-value $w_{i_1 \ldots i_m}$ corresponds to the weight of assigning the same color to m edges $e_1^{i_1} = (1, \widetilde{v}_1^{i_1})$, $e_2^{i_2} = (2, \widetilde{v}_2^{i_2})$, ..., $e_m^{i_m} = (m, \widetilde{v}_m^{i_m})$,

$$w_{i_1 \ldots i_m} = \begin{cases} \max\limits_{1 \le h \le m} \{w(e_h^{i_h})\}, & \text{if all vertices } \widetilde{v}_1^{i_1}, \widetilde{v}_2^{i_2}, \ldots, \widetilde{v}_m^{i_m} \text{ are different,} \\ \infty, & \text{otherwise.} \end{cases} \tag{15}$$

Again, the array with entries $\max_{1 \le h \le m}\{w(e_h^{i_h})\}$ satisfies the Monge property, see (5). Therefore, the array W defined by (15), where ∞-entries correspond to incompatible pairs of edges, is a nearly Monge array with $\lambda = 1$ incompatible partner for every $i_h = i_h^*$. Thus, similar to the two-dimensional case, we have

$$\text{MEC}^{mn}(K_{m,mn}) \propto \text{AP}(m, 1), \tag{16}$$

which together with (14) proves the theorem. □

In the proof of Theorem 1 we have developed reduction (11), which is useful for handling the generalized versions of the two applications, scheduling satellite transmissions and synchronous open shop problems. These two problems were introduced in Section 2 under the assumption that exactly d activities should be assigned in each period/cycle. Both problems were modeled as AP(d, 1) by defining for each sender or machine ℓ, $1 \le \ell \le d$, the n-tuple $\Gamma^\ell = (\gamma_1^\ell, \gamma_2^\ell, \ldots, \gamma_n^\ell)$, where γ-values are in the non-decreasing order (6), and by setting up the cost array $\hat{W}$ of the form:

$$\hat{w}_{i_1 \ldots i_d} = \begin{cases} \max\{\gamma_{i_1}^1, \gamma_{i_2}^2, \ldots, \gamma_{i_d}^d\}, & \text{if all activities corresponding to } \gamma_{i_1}^1, \ldots, \gamma_{i_d}^d \text{ are compatible,} \\ \infty, & \text{otherwise,} \end{cases} \tag{17}$$

see (7) and (8). If fewer than d activities per period/cycle are allowed, then using Observation 1 and reduction (14) together with (16), the two applications are modeled as AP(d, 1) with n actual receivers/jobs and $(d-1)n$ auxiliary ones. We set the γ-values for the auxiliary receivers/jobs to 0. Then each extended dn-tuple Γ^ℓ is of the form

$$\Gamma^\ell = \left(\underbrace{0, 0, \ldots, 0}_{d(n-1) \text{ elements}}, \underbrace{\gamma_1^\ell, \gamma_2^\ell, \ldots, \gamma_n^\ell}_{n \text{ elements}} \right).$$

This results in the extended cost array $\widetilde{W}$ with entries of type (17) defined for dn receivers/jobs rather than for n. If parameters d and λ are fixed, then the $O(n)$-time algorithm of Section 6 solves the associated applied problems in linear time, after sorting the input data in order to achieve (6).

We now turn to complexity aspects of problem AP(d, λ) with one parameter fixed and another one being part of the input. If $\lambda = 0$, then the weight array of problem AP(d, 0) is Monge rather than nearly Monge, and an optimal solution is of the diagonal structure [5]. For the case of $\lambda = 1$ we use Theorem 1 and the NP-hardness result known for MEC($K_{m,n}$) from [10,27] to make the following conclusion.

Observation 2. *Problem* AP(d, 1)*, where* $d \geq 2$ *is part of the input, is NP-hard in the strong sense, even if there are only three distinct finite weights.*

Note that the restriction related to three weight values, except for ∞'s, follows from the reduction presented in [27], which uses only three distinct weights. Interestingly, problem MEC($K_{n,n}$) with two distinct weights is solvable in polynomial time, as shown in [10,27].

Consider now the counterpart of AP(d, λ) with fixed d and an arbitrary λ. If $d = 2$, then problem AP(2, λ) can be solved in $O(n^3)$ time by the Hungarian algorithm [4,21] that finds an optimal solution to the assignment problem with an arbitrary, not necessarily Monge-like matrix. If $d = 3$, then problem AP(3, λ) becomes NP-hard in the strong sense.

Observation 3. *Problem* AP(3, λ) *is NP-hard in the strong sense, even if all finite weights are equal.*

To justify Observation 3 we establish a link between problem AP(3, λ) and the 3-DIMENSIONAL MATCHING problem (3-DM) known to be strongly NP-complete. Recall that in 3-DM, there are given three disjoint sets X, Y and Z, with n elements in each, and a set of triples $M \subseteq (X \times Y \times Z)$; the goal is to find a perfect matching, i.e. a set $M' \subseteq M$, with $|M'| = n$ such that no element from X, Y or Z appears in more than one triple in M', see [14]. Any instance of problem AP(3, λ) with all finite weights equal to 0 can be treated as an instance of problem 3-DM, where the set M consists of all triples excluding those with incompatible partner indices. We denote this special case of 3-DM by 3-DM(λ). Conversely, any instance of problem 3-DM(λ) can be treated as an instance of problem AP(3, λ) where all finite weights are 0. Clearly, there exists a perfect matching for an instance of 3-DM(λ) if and only if there exists an assignment of cost 0 for the corresponding instance of AP(3, λ). Thus,

$$\text{3-DM}(\lambda) \propto \text{AP}(3, \lambda).$$

In order to demonstrate the NP-completeness of 3-DM(λ), consider the known NP-completeness proof for 3-DM. It is based on the reduction 3-SAT$\propto$ 3-DM presented, e.g., in [14]. Without changing the structure of the instance of 3-DM used in the proof, one can re-define it as an instance of 3-DM(λ) with $\lambda = n - 1$, so that the reduction from [14] becomes 3-SAT$\propto$ 3-DM(λ), see the Online Supplement to this paper (Appendix A). for an example of such a re-definition. Thus, we get the chain of reductions

$$\text{3-SAT} \propto \text{3-DM}(\lambda) \propto \text{AP}(3, \lambda),$$

which proves Observation 3.

The complexity results discussed in this section are summarized in Table 1 in the Conclusions.

4. Some properties of nearly Monge matrices with incompatible partner indices

In this and in the subsequent sections we consider problem AP(d, λ) with fixed d and λ. Prior to discussing algorithmic aspects, we first demonstrate in Section 4.1 that the existing results known for the assignment problem with ∞'s are generally inapplicable. Then in Sections 4.2–4.3 we address several typical questions that arise in the context of matrices with ∞'s or with unspecified entries.

4.1. Nonexistence of a Monge sequence

As explained before, the introduction of ∞-entries into a Monge matrix may destroy the Monge property. In the literature on Monge-like structures it is then suggested to verify whether a non-Monge matrix possesses a so-called Monge sequence. Such a sequence guides greedy algorithm towards finding an optimal solution to the assignment or transportation problem [5,18]. It considers the variables in the order they appear in a Monge sequence and assigns the highest possible values to them without violating the constraints of the problem. Formally, a *Monge sequence* of an $n \times n$ matrix $W = (w_{ij})$ is defined as a sequence of n^2 index pairs $(i_1, j_1), \ldots, (i_{n^2}, j_{n^2})$ such that whenever (i, j) precedes both (i, s) and (r, j) in the sequence, condition

$$w_{ij} + w_{rs} \leq w_{is} + w_{rj} \tag{18}$$

holds (cf. [18]).

A Monge sequence, if one exists, is a powerful tool for solving problems with matrices which do not satisfy the Monge property for all quadruples. Models of this type arise for example in the scheduling context. The scheduling problems studied

in [16,17] can be modeled as transportation problems with non-Monge matrices. In some cases Monge sequences can be found efficiently; in others a Monge sequence does not exist, and this calls for a development of special tailor-made algorithms, as in the case of the generalized (weighted) version of the problem from [17].

Even for the simplest version of our problem AP(d, λ) with $d = 2$ and $\lambda = 1$ the matrices in general do not possess a Monge sequence. Consider the two applications discussed in Section 2 with a cost matrix of type (5). If the w-values are defined as

$$w_{ij} = \begin{cases} \max\{i, j\}, & \text{if } i \neq j, \\ \infty, & \text{otherwise,} \end{cases}$$

and the matrix is at least of size 3×3, then a Monge sequence does not exist. Indeed, whichever element is selected as the first element of a Monge sequence, the sequence cannot be completed to satisfy (18). For any selected entry there always exists an ∞-entry that is not in the same row and not in the same column, while the other two entries of the associated quadruple are finite. Clearly, condition (18) is violated for such a quadruple. For a more general case, a similar example can be found in [30], where finite entries of W are arbitrary.

Observation 4. *In general, nearly Monge arrays with incompatible partners do not give rise to Monge sequences even for $\lambda = 1$ and $d = 2$.*

4.2. Recognizing nearly Monge arrays

Given a weight matrix W, it is easy to verify whether the Monge property (2) is satisfied for finite entries. A straightforward way to do so is to check (2) for all pairs of finite entries of the form (w_{ij}, w_{rs}) with $i < r$ and $j < s$, which takes no more than $O(n^4)$ checks. This method can also be generalized to recognize d-dimensional nearly Monge arrays, in which case it takes no more than $O(n^{2d})$ checks. Note that a d-dimensional nearly Monge array consists of n^d entries.

The applications discussed in Section 2 and the MEC problem of Section 3 deal with a weight array W of form (5) with ∞-entries introduced for incompatible partners. Even if initially W is not nearly Monge, that property can be achieved by ordering the γ-values of lists Γ^ℓ, which is equivalent to sequencing the sets I and J of the assignment problem, or permuting the rows and columns of W. In general, however, recognizing a permuted nearly Monge array is a non-trivial task, as discussed below. Array W is a *permuted* nearly Monge array if its index sets can be permuted to make the array nearly Monge.

Observation 5. *For an arbitrary λ, it is NP-complete to decide whether a given array with at most λ incompatible partners (for every index) is permuted nearly Monge, even for $d = 2$ (i.e. the matrix case).*

This observation follows from a similar result in [9] formulated for a permuted incomplete Monge matrix, which is equivalent to a nearly Monge matrix with an arbitrary λ. Notice that recognizing a permuted Monge matrix can be done in $O(n^2)$ time [5], and recognizing a special incomplete Monge matrix, namely a Supnick matrix, which is a symmetric Monge matrix with unspecified diagonal entries, can be done in $O(n^2 \log n)$ time [9].

In the nearly Monge case, if the number of incompatible partners λ is fixed, recognition of permuted nearly Monge matrices remains open. If a polynomial-time algorithm could be developed, it would be beneficial to achieve a time complexity smaller than $O(n^3)$, beating the time complexity of solving a general assignment problem.

Further difficulties arise in recognition of nearly Monge arrays for higher dimensions $d \geq 3$. It is known that in the absence of ∞'s, a d-dimensional array W is a Monge array if and only if every two-dimensional submatrix is a Monge matrix [1]. This property can then be efficiently used in recognizing d-dimensional Monge arrays [29]. Unfortunately this necessary and sufficient condition no longer holds for nearly Monge arrays, even if $\lambda = 1$. Consider for example a three-dimensional array $W = (w_{ijk})$ with $n = 3$, $\lambda = 1$, incompatible partners $(1, 2, *)$, $(2, 1, *)$, $(1, *, 3)$, $(2, *, 2)$, $(3, *, 1)$, $(*, 1, 2)$, $(*, 2, 3)$, two 1-entries $w_{111} = w_{333} = 1$ and all remaining finite entries being 0's. The two-dimensional submatrices are listed below, and all of them are nearly Monge. However, because of $w_{111} + w_{333} = 2 > 0 = w_{131} + w_{313}$, the array W is not a nearly Monge array.

$$i = 1: \begin{pmatrix} 1 & \infty & \infty \\ \infty & \infty & \infty \\ 0 & 0 & \infty \end{pmatrix} \quad i = 2: \begin{pmatrix} \infty & \infty & \infty \\ 0 & \infty & \infty \\ 0 & \infty & 0 \end{pmatrix} \quad i = 3: \begin{pmatrix} \infty & \infty & 0 \\ \infty & 0 & \infty \\ \infty & 0 & 1 \end{pmatrix}$$

$$j = 1: \begin{pmatrix} 1 & \infty & \infty \\ \infty & \infty & \infty \\ \infty & \infty & 0 \end{pmatrix} \quad j = 2: \begin{pmatrix} \infty & \infty & \infty \\ 0 & \infty & \infty \\ \infty & 0 & \infty \end{pmatrix} \quad j = 3: \begin{pmatrix} 0 & 0 & \infty \\ 0 & \infty & 0 \\ \infty & 0 & 1 \end{pmatrix}$$

$$k = 1: \begin{pmatrix} 1 & \infty & 0 \\ \infty & 0 & 0 \\ \infty & \infty & \infty \end{pmatrix} \quad k = 2: \begin{pmatrix} \infty & \infty & 0 \\ \infty & \infty & \infty \\ \infty & 0 & 0 \end{pmatrix} \quad k = 3: \begin{pmatrix} \infty & \infty & \infty \\ \infty & \infty & 0 \\ 0 & \infty & 1 \end{pmatrix}$$

Observation 6. *Even if every two-dimensional submatrix of a d-dimensional array W with $\lambda = 1$ is a nearly Monge matrix, the whole array W is not necessarily a nearly Monge array.*

4.3. Completing nearly Monge arrays

An interesting question related to arrays with unspecified or ∞-entries is the possibility of completing them by introducing finite values in order to achieve Monge arrays. Such an approach works, for example, for incomplete matrices of Supnick type, as shown in [9], but unfortunately, it does not work for nearly Monge matrices. Consider the following nearly Monge matrix with $\lambda = 1$:

$$\begin{pmatrix} 0 & 0 & 1 & \infty & 0 \\ 1 & 0 & 0 & 0 & \infty \\ 4 & 2 & \infty & 0 & 1 \\ 6 & \infty & 2 & 0 & 0 \\ \infty & 6 & 4 & 1 & 0 \end{pmatrix}.$$

Attempting to complete that matrix without permuting rows and columns, we have to satisfy the following two contradicting conditions for w_{14}:

$$\begin{aligned} w_{14} + w_{23} &\geq w_{13} + w_{24}, \quad \text{or equivalently } w_{14} \geq 1, \\ w_{14} + w_{35} &\leq w_{15} + w_{34}, \quad \text{or equivalently } w_{14} \leq -1. \end{aligned}$$

Attempting to permute and complete the above matrix, the only suitable permutation of rows and columns that preserves the Monge property for finite entries is the one which reverses the orders of rows and columns (notice that all quadruples with finite entries satisfy the Monge property as strict inequalities). The arguments for entry w_{14} can be re-used with respect to the entry w_{52} in the permuted matrix, justifying that completing cannot be done.

Observation 7. *In general, a nearly Monge matrix with incompatible partners cannot be completed into a Monge matrix by replacing ∞-entries by finite values even for $\lambda = 1$.*

If permutations of rows and columns are fixed, producing a completed Monge matrix, if one exists, can be done in polynomial time by solving the following system of linear inequalities:

$$\begin{aligned} \hat{w}_{ij} + \hat{w}_{i+1,j+1} &\leq \hat{w}_{i,j+1} + \hat{w}_{i+1,j}, && 1 \leq i, j < n, \\ \hat{w}_{ij} &= w_{ij}, && 1 \leq i, j < n,\ w_{ij} \neq \infty, \\ \hat{w}_{ij} &\in \mathbb{R}. \end{aligned}$$

Here $\hat{w}_{ij}$ are real-valued variables representing the target values of w_{ij}. Notice that for a Monge matrix it is sufficient to achieve the Monge property for quadruples defined by adjacent pairs of rows and columns [5].

If a feasible solution satisfying the above inequalities exists, real values are assigned to all ∞-entries, so that the resulting Monge matrix $\hat{W}$ is a completion of W. In the case of infeasibility, either the matrix W is not nearly Monge (i.e. not all quadruples of finite entries satisfy the Monge property) or the matrix W is not completable to a Monge matrix.

Notice that in the applications, which arise in scheduling satellite transmissions or synchronous open shops, nearly Monge arrays are completable if the formulae for w's (7) and (8) are modified accordingly, namely, by assigning finite values $\max\left\{\gamma^1_{i_1}, \gamma^2_{i_2}, \ldots, \gamma^d_{i_d}\right\}$ to all d-tuples $(i_1, i_2, \ldots, i_n)$, ignoring the condition that all receivers or all jobs should be different. Also, formula (9) for MEC $\left(K_{m,n}\right)$ can be adjusted to replace ∞'s by values $\max\{w(e)|e \in E_c\}$ for any combinations of m edges with different origins in V_1, even if there are repeated vertices in V_2. Due to the sorting and the max-definition (5), the resulting array is Monge.

5. The corridor property for problem AP(d, λ)

In this section we consider the assignment problem AP(d, λ) with a d-dimensional nearly Monge weight array W and at most λ incompatible partners for every index. We show that there exists an optimal solution $\hat{S}$ such that all non-zero entries $x_{i_1,\ldots,i_d} = 1$ of the solution matrix $X_{\hat{S}}$ lie in a *corridor* of a certain width ξ around the main diagonal,

$$|i_\ell - i_1| \leq \xi \quad \text{for all } \ell = 1, \ldots, d, \tag{19}$$

where

$$\xi = d(d-1)\lambda. \tag{20}$$

We refer to the above condition as the *corridor property*. Based on it, in the next section we develop a linear-time algorithm to solve problem AP(d, λ) for fixed d and λ. Note that AP(d, λ) is strongly NP-hard, if d or λ is part of the input, see Section 3.

Theorem 2. *For problem* AP(d, λ) *there exists an optimal solution* $\hat{S} = \{(\hat{i}_1^1, \hat{i}_2^1, \ldots, \hat{i}_d^1), \ldots, (\hat{i}_1^n, \hat{i}_2^n, \ldots, \hat{i}_d^n)\}$ *such that every* d*-tuple* $(i_1, i_2, \ldots, i_d) = (\hat{i}_1^k, \hat{i}_2^k, \ldots, \hat{i}_d^k)$, $1 \le k \le n$, *satisfies* (19) *with* ξ *defined by* (20).

Proof. Starting with an optimal solution $S = \left\{(i_1^1, i_2^1, \ldots, i_d^1), \ldots, (i_1^n, i_2^n \ldots, i_d^n)\right\}$ that violates the corridor property (19) we perform a series of exchange steps, each of which does not increase the cost, eventually producing a solution that satisfies (19). We assume that $w(S) < \infty$, i.e., none of the d-tuples in S contains a pair of incompatible partners; otherwise S can be replaced by the diagonal solution, which has the desired property and no higher cost.

For the exchange step, we take a d-tuple $(i_1, i_2, \ldots, i_d)$ of the current solution that violates (19), select a special companion d-tuple $(j_1, j_2, \ldots, j_d)$ from the current solution and replace that pair by $(s_1, s_2, \ldots, s_d)$ and $(t_1, t_2, \ldots, t_d)$ defined by

$$s_\ell = \min\{i_\ell, j_\ell\}, \qquad t_\ell = \max\{i_\ell, j_\ell\}, \qquad \ell = 1, \ldots, d. \tag{21}$$

Since W is nearly Monge, the inequality

$$w_{s_1 s_2 \ldots s_d} + w_{t_1 t_2 \ldots t_d} \le w_{i_1 i_2 \ldots i_d} + w_{j_1 j_2 \ldots j_d}$$

holds, if none of the four entries is ∞. With an appropriate choice of the companion d-tuple $(j_1, j_2, \ldots, j_d)$, we eliminate the violation related to $(i_1, i_2, \ldots, i_d)$ without increasing the cost.

We distinguish between two types of violations:

$$\begin{aligned} &\text{Type I}(i_\ell) \text{ violation}: \quad i_\ell > i_1 + \xi \quad \text{for } i_1 \le n - \xi - 1, \\ &\text{Type II}(i_\ell) \text{ violation}: \quad i_\ell < i_1 - \xi \quad \text{for } i_1 \ge \xi + 1, \end{aligned}$$

and use the terms "d-tuple of Type I(i_ℓ)" and "d-tuple of Type II(i_ℓ)" for violating d-tuples.

First we eliminate Type I(i_2) violations for the second index. We start with the d-tuple of Type I(i_2) with the smallest first index i_1 and then we proceed with other d-tuples of Type I(i_2) considering them in increasing order of i_1. After that we eliminate all Type II(i_2) violations, starting with the d-tuple of Type II(i_2) with the largest first index i_1, proceeding then with other d-tuples of Type II(i_2) considered in decreasing order of i_1.

Having eliminated all Type I(i_2) and Type II(i_2) violations for the second index, we proceed with violations for the third index. We handle them in the same order, fixing first Type I(i_3) violations with d-tuples of Type I(i_3) considered in increasing order of i_1, and then Type II(i_3) violations with d-tuples of Type II(i_3) considered in decreasing order of i_1. For these exchanges we demonstrate that no new violations of Type I(i_2) and Type II(i_2) for second indices are created. The same approach is then applied to the remaining indices $i_4, \ldots, i_d$.

Eliminating violations of Type I(i_2). Among all d-tuples with violations of Type I(i_2), select the d-tuple $\left(\underline{i_1}, \underline{i_2}, \ldots, \underline{i_d}\right)$ with the smallest first index $\underline{i_1}$, so that

$$\underline{i_2} > \underline{i_1} + \xi. \tag{22}$$

Note that the choice of $\underline{i_1}$ implies that no Type I(i_2) violations happen for $i_1 < \underline{i_1}$. For the violating d-tuple $\left(\underline{i_1}, \underline{i_2}, \ldots, \underline{i_d}\right)$ select a companion d-tuple $(j_1, j_2, \ldots, j_d)$ with

$$j_1 > \underline{i_1} \quad \text{and} \quad j_2 \le \underline{i_1} + \xi \tag{23}$$

such that

(a) $x_{j_1 j_2 \ldots j_d} = 1$ in the current solution (recall that the associated value of $w_{j_1 j_2 \ldots j_d}$ cannot be ∞ then),

(b) the induced d-tuples $(s_1, s_2, \ldots, s_d)$, $(t_1, t_2, \ldots, t_d)$ defined by (21) correspond to finite entries in W, i.e. do not contain two incompatible partner indices.

For violation (22) the required d-tuple $(j_1, j_2, \ldots, j_d)$ exists since, as we show below, there are $\xi + 1$ candidate 1-entries with indices j_1 and j_2 satisfying (23). Clearly, at least one of those $\xi + 1$ candidates has all indices compatible with the indices from $\left(\underline{i_1}, \underline{i_2}, \ldots, \underline{i_d}\right)$. If this was not the case, i.e., if there was at least one incompatible index between $\left(\underline{i_1}, \underline{i_2}, \ldots, \underline{i_d}\right)$ and each of the $\xi + 1$ candidate entries, then the total number of incompatible partner indices for $\left(\underline{i_1}, \underline{i_2}, \ldots, \underline{i_d}\right)$ would be equal to $\xi + 1 = d(d-1)\lambda + 1$, a contradiction to the assumption that for any index $i_u = \underline{i_u}$, $1 \le u \le d$, there are at most $\Omega = \lambda(d-1)$ incompatible partners, see (4).

We now justify that there are indeed $\xi + 1$ candidate 1-entries with indices j_1 and j_2 satisfying (23). We start with the 2-dimensional case, with the binary solution matrix $X_S = (x_{i_1 i_2})$ illustrated in Fig. 2(a). The symbol "×" in Fig. 2 is used to mark compulsory 0's in the solution matrix, caused by incompatible partner indices.

Consider the first $(\underline{i_1} + \xi)$ columns of the associated matrix X_S, marked by a rounded rectangle in Fig. 2(a). Since every column of X_S contains one 1-entry, the selected part contains $(\underline{i_1} + \xi)$ 1-entries. By the assumption, there are no violations of Type I(i_2) for the first $(\underline{i_1} - 1)$ rows; therefore the first $(\underline{i_1} - 1)$ rows of the selected part contain $(\underline{i_1} - 1)$ 1-entries in columns $1, 2, \ldots, \underline{i_1} + \xi - 1$, and they do not contain 1-entries in column $\underline{i_1} + \xi$. Thus, the remaining part of the selection, corresponding to rows $\underline{i_1} + 1, \ldots, n$ and columns $1, 2, \ldots, \underline{i_1} + \xi$, contains

$$\left(\underline{i_1} + \xi\right) - \left(\underline{i_1} - 1\right) = \xi + 1$$

1-entries.

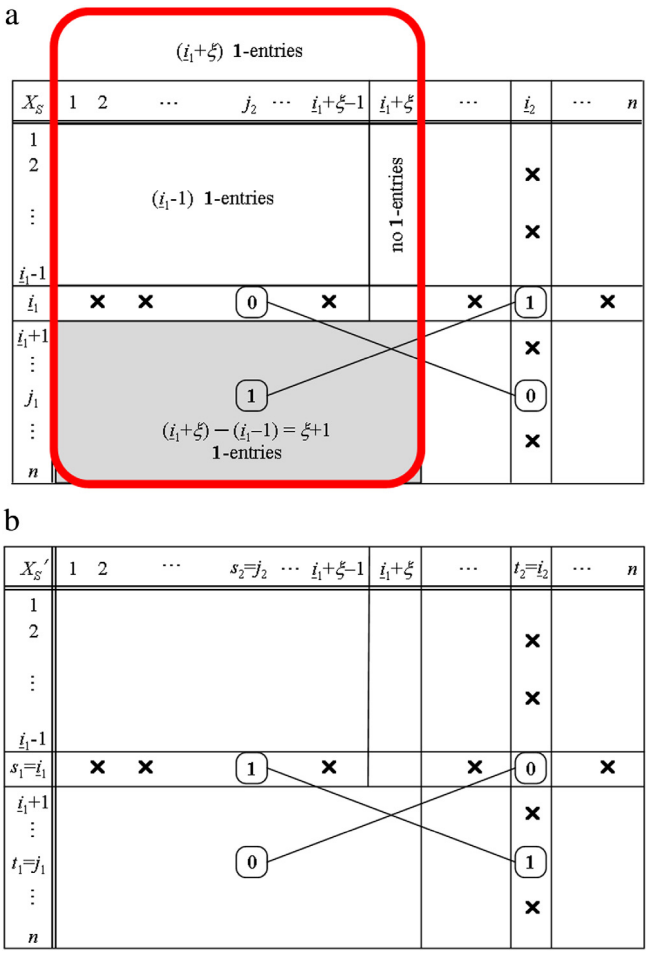

Fig. 2. Eliminating Type I(i_2) violation.
(a) Solution matrix X_S with the violating d-tuple $(\underline{i}_1, \underline{i}_2, \ldots, \underline{i}_d)$ of Type I(i_2).
(b) Modified solution matrix X_S' with Type I(i_2) violation in row $\underline{i}_1$ eliminated.

It is easy to see that the same arguments are applicable for the multi-dimensional case. Fig. 2 can be treated as a 'projection' of the d-dimensional case into the space of the first two indices with

$$\widetilde{x}_{i_1 i_2} = \sum_{i_3,\ldots,i_d} x_{i_1 i_2 i_3 \ldots i_d}.$$

Replacing $(\underline{i}_1, \underline{i}_2, \ldots, \underline{i}_d)$, $(j_1, j_2, \ldots, j_d)$ by $(s_1, s_2, \ldots, s_d)$, $(t_1, t_2, \ldots, t_d)$ eliminates the Type I(i_2) violation for $i_1 = \underline{i}_1$. Note that it does not matter whether the companion d-tuple $(j_1, j_2, \ldots, j_d)$ is violating or not; that d-tuple is removed from the solution as a result of the exchange step. It also does not matter whether $(s_1, s_2, s_3, \ldots, s_d) = (\underline{i}_1, j_2, s_3, \ldots, s_d)$ or $(t_1, t_2, \ldots, t_d) = (j_1, \underline{i}_2, t_3, \ldots, t_d)$ are violating: a possible violation for $(s_1, s_2, \ldots, s_d)$ can only be of Type II(i_2), as $s_2 = j_2 \le \underline{i}_1 + \xi$ by (23), and it is repaired at a later stage; a possible violation for $(t_1, t_2, \ldots, t_d)$ may be of any type, but due to $t_1 = j_1 > \underline{i}_1$ it is also repaired at a later stage.

Proceeding with each next smallest index i_1, for which violation of Type I(i_2) occurs, all Type I(i_2) violations are repaired in a similar way, resulting in a solution where no Type I(i_2) violations remain.

Eliminating violations of Type II(i_2). Among all d-tuples with violations of Type II(i_2), select the d-tuple $(\overline{i}_1, \overline{i}_2, \overline{i}_3, \ldots, \overline{i}_d)$ with the largest first index $\overline{i}_1$, so that the corridor property is satisfied for the second index for any $i_1 > \overline{i}_1$ and the violation under consideration is of the form

$$\overline{i}_2 < \overline{i}_1 - \xi, \tag{24}$$

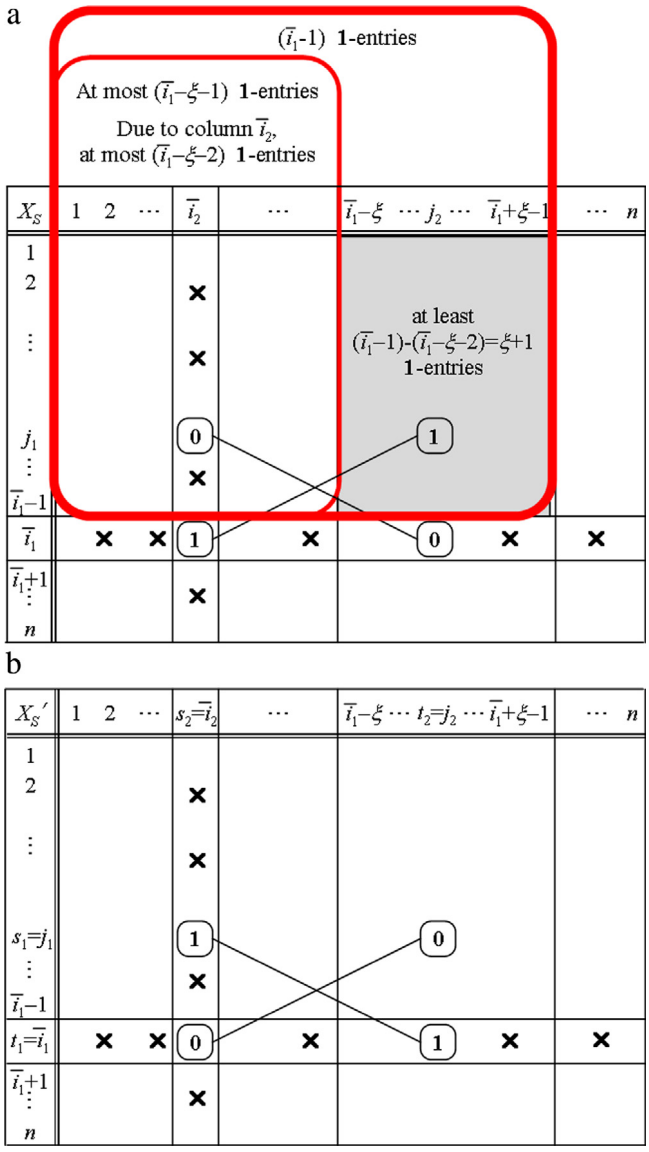

Fig. 3. Eliminating Type II(i_2) violation.
(a) Solution matrix X_S with the violating d-tuple $(\overline{i}_1, \overline{i}_2, \ldots, \overline{i}_d)$ of Type II(i_2).
(b) Modified solution matrix X_S' with Type II(i_2) violation in row $\overline{i}_1$ eliminated.

see Fig. 3. The required companion d-tuple $(j_1, j_2, \ldots, j_d)$ should satisfy

$$j_1 < \overline{i}_1 \quad \text{and} \quad \overline{i}_1 - \xi \le j_2 \le \overline{i}_1 + \xi, \tag{25}$$

together with properties (a)–(b) formulated for Type I(i_2) violations.

For violation (24) the required d-tuple $(j_1, j_2, \ldots, j_d)$ exists since, as we show below, there are $\xi + 1$ candidate 1-entries with indices j_1 and j_2 satisfying

$$j_1 < \overline{i}_1 \quad \text{and} \quad \overline{i}_1 - \xi \le j_2 \le \overline{i}_1 + \xi - 1, \tag{26}$$

which is in fact a stronger condition than (25). Again, at least one of those $\xi + 1$ entries has all indices compatible with the indices from $(\overline{i}_1, \overline{i}_2, \ldots, \overline{i}_d)$, so that the exchange step with the induced d-tuples $(s_1, s_2, \ldots, s_d)$, $(t_1, t_2, \ldots, t_d)$ achieves its goal.

To justify that there exist $\xi + 1$ candidate 1-entries with indices j_1 and j_2 satisfying (26), consider first the 2-dimensional case. Since X_S does not contain Type I(i_2) violations, all $(\overline{i}_1 - 1)$ 1-entries of the $(\overline{i}_1 - 1)$ first rows belong to the area marked by the larger rounded rectangle in Fig. 3(a) with $j_2 \le \overline{i}_1 + \xi - 1$. Moreover, the part of the selected area $j_2 \le \overline{i}_1 - \xi - 1$,

marked by the smaller rounded rectangle in Fig. 3(a), contains no more than $\left(\overline{i_1} - \xi - 1\right)$ 1-entries. In fact it contains no more than $\left(\overline{i_1} - \xi - 2\right)$ 1-entries since the 1-entry in column $\overline{i_2}$ does not belong to the selected area. Thus, the remaining part of the selection, corresponding to rows $1, 2, \ldots, \overline{i_1} - 1$ and columns $\overline{i_1} - \xi, \ldots, \overline{i_1} + \xi - 1$ contains

$$\left(\overline{i_1} - 1\right) - \left(\overline{i_1} - \xi - 2\right) = \xi + 1$$

1-entries, which all satisfy (26). Again it is easy to see that the same arguments hold for the d-dimensional case.

Replacing $\left(\overline{i_1}, \overline{i_2}, \ldots, \overline{i_d}\right)$, $(j_1, j_2, \ldots, j_d)$ by $(s_1, s_2, \ldots, s_d)$, $(t_1, t_2, \ldots, t_d)$ eliminates violation (24), so that the corridor property is now achieved for the second index for any $i_1 \geq \overline{i_1}$. Note that $(t_1, t_2, \ldots, t_d) = \left(\overline{i_1}, j_2, t_3, \ldots, t_d\right)$ satisfies the corridor property, while $(s_1, s_2, s_3, \ldots, s_d) = \left(j_1, \overline{i_2}, s_3, \ldots, s_d\right)$ may violate it. The possible violation is of Type II(i_2) with $s_1 < \overline{i_1}$, and it is repaired at a later stage.

Proceeding with each next largest index i_1, for which a violation of Type II(i_2) occurs, all Type II(i_2) violations are repaired in a similar way, resulting in a solution where no violations of any type remain for the second index.

After all violations of Type I(i_2) and Type II(i_2) are eliminated, we start eliminating violations for the third index, ensuring that no new violations are created for the second index. We use the same approach:

- first eliminate all violations of Type I(i_3), starting with a violating d-tuple with the smallest i_1, and proceeding then with other violating d-tuples considered in increasing order of i_1;
- next eliminate all violations of Type II(i_3), starting with a violating d-tuple with the largest i_1, and proceeding with other violating d-tuples considered in decreasing order of i_1.

The existence of the d-tuple $(j_1, j_2, \ldots, j_d)$ needed for the exchange step can be proven in the same way as above. We only need to demonstrate that the induced d-tuples $(s_1, s_2, \ldots, s_d)$, $(t_1, t_2, \ldots, t_d)$, defined in accordance with (21), do not create new violations in the previously repaired second index, while repairing violations in the third one.

Consider the exchange step based on $(i_1, i_2, \ldots, i_d)$ and $(j_1, j_2, \ldots, j_d)$ with

$$i_1 < j_1. \tag{27}$$

Since in this stage violations for the second index have been already repaired, we have

$$-\xi \leq i_2 - i_1 \leq \xi, \tag{28}$$
$$-\xi \leq j_2 - j_1 \leq \xi. \tag{29}$$

If $i_2 < j_2$, then the induced d-tuples are of the form $(i_1, i_2, s_3 \ldots, s_d)$, $(j_1, j_2, t_3 \ldots, t_d)$, and by (28)–(29) no new violation appears for the second index. Alternatively, if

$$i_2 > j_2, \tag{30}$$

then for the induced d-tuple $(i_1, j_2, s_3 \ldots, s_d)$ there is no violation for the second index since

$j_2 - i_1 > -\xi$ by (27) and by the first inequality from (29),
$j_2 - i_1 < \xi$ by (30) and by the second inequality from (28).

For the induced d-tuple $(j_1, i_2, s_3 \ldots, s_d)$ there is also no violation for the second index since

$i_2 - j_1 > -\xi$ by (30) and by the first inequality from (29),
$i_2 - j_1 < \xi$ by (27) and by the second inequality from (28).

It is easy to verify that if instead of (27) condition $i_1 > j_1$ holds, then similar arguments are applicable. Thus, repairing violations related to the third index in the described way cannot create violations related to the second one.

Using the same approach repeatedly, we eliminate violations with respect to each index i_ℓ, $\ell = 4, \ldots, d$. Each time, when eliminating violations for i_ℓ, we do not create new violations for indices $i_2, i_3, \ldots, i_{\ell-1}$. This completes the proof of Theorem 2. □

6. A linear-time algorithm for problem AP(d, λ) with fixed d and λ

In this section we consider problem AP(d, λ) with fixed d and λ and develop a linear-time algorithm for it. We start with the two-dimensional problem AP(2, λ).

By Theorem 2, we can restrict the search to solutions S such that all 1-entries appear inside the corridor, which can be considered as a combination of the main diagonal, 2λ diagonals above it and 2λ diagonals below it. An example of a solution matrix X_S that satisfies the described property is presented below for $\lambda = 1$, $d = 2$ and $n = 10$, with the corridor for 1-entries marked by $*$:

X_S	1	2	3	4	5	6	7	8	9	10
1	*	*	*	0	0	0	0	0	0	0
2	*	*	*	*	0	0	0	0	0	0
3	*	*	*	*	*	0	0	0	0	0
4	0	*	*	*	*	*	0	0	0	0
5	0	0	*	*	*	*	*	0	0	0
6	0	0	0	*	*	*	*	*	0	0
7	0	0	0	0	*	*	*	*	*	0
8	0	0	0	0	0	*	*	*	*	*
9	0	0	0	0	0	0	*	*	*	*
10	0	0	0	0	0	0	0	*	*	*

Introduce a layered network $L = (s, t, V, A)$ with the vertex set V consisting of disjoint subsets $V(0)$, $V(1), \ldots, V(n)$, $V(n+1)$. Subsets $V(0)$ and $V(n+1)$ are single-element subsets containing the source $V(0) = \{s\}$ and the sink $V(n+1) = \{t\}$. The arcs A have end vertices belonging to two consecutive layers, $A = \cup_{k=0}^{n} (V(k) \times V(k+1))$. The vertices of layer $V(k)$, $1 \le k \le n$, characterize the partial solutions consisting of the first k rows and satisfying the corridor property.

For two vertices $v_\ell(k) \in V(k)$ and $v_m(k+1) \in V(k+1)$, let $X_\ell(k)$ and $X_m(k+1)$ be two associated partial solutions. If $X_\ell(k) \subset X_m(k+1)$, then the arc $(v_\ell(k), v_m(k+1))$ has the cost of assigning the 1-entry to a relevant position in row $k+1$; that cost is given by $w_{k+1,j}$, where j is the column with $x_{k+1,j} = 1$ in $X_m(k+1)$. Otherwise $v_\ell(k)$ and $v_m(k+1)$ are not connected by an arc. For completeness, introduce auxiliary arcs of cost 0 from every vertex of $V(n)$ to t. Clearly, an optimal solution corresponds to a shortest path in the network L.

In order to reduce the size of L, we include in $V(k)$ only the vertices corresponding to non-dominated partial solutions: if $X_{\ell'}(k)$ and $X_{\ell''}(k)$ have 1-entries assigned to the same set of columns and $w(X_{\ell'}(k)) \le w(X_{\ell''}(k))$, where $w(X)$ represents the cost of the associated solution, then it is sufficient to include in $V(k)$ only one vertex corresponding to $X_{\ell'}(k)$. This implies that each vertex of $V(k)$ is characterized by a unique subset of columns which contain 1-entries in the first k rows. In order to estimate $|V(k)|$, we prove the following statement.

Statement 1. *Any partial solution $X_\ell(k)$ of layer k consists of*

(a) *α columns $1, 2, \ldots, \alpha$, each containing one 1-entry, where*

$$\alpha = \max\{k - 2\lambda, 0\},$$

(b) *$n - \beta$ columns $\beta + 1, \ldots, n$, each containing 0-entries only, where*

$$\beta = \min\{k + 2\lambda, n\},$$

(c) *among the columns $\alpha + 1, \ldots, \beta$ there are $(k - \alpha) = \min\{2\lambda, k\}$ columns, each containing one 1-entry, and $(\beta - k) = \min\{2\lambda, n - k\}$ columns with 0-entries only.*

The following example with $k = 6$, $n = 10$, $\lambda = 1$ illustrates the structure of a partial solution $X_\ell(k)$:

		α columns with 1-entry each				$(k-\alpha)$ columns with 1-entries $(\beta-k)$ columns without 1-entries				no 1-entries	
					α ↓	$\alpha+1$ ↓	k ↓		β ↓	$\beta+1$ ↓	n ↓
	$X_\ell(k)$	1	2	3	4	5	6	7	8	9	10
	1	*	*	*	0	0	0	0	0	0	0
	2	*	*	*	*	0	0	0	0	0	0
	3	*	*	*	*	*	0	0	0	0	0
	4	0	*	*	*	*	*	0	0	0	0
	5	0	0	*	*	*	*	*	0	0	0
$k \to$	6	0	0	0	*	*	*	*	*	0	0
$k+1 \to$	7	0	0	0	0	*	*	*	*	*	0

Proof. Conditions (a) and (b) hold since by the corridor property there can be no 1-entry for any combination of $i \in \{k+1, \ldots, n\}$ and $j \in \{1, 2, \ldots, \alpha\}$, and also for any combination of $i \in \{1, 2, \ldots, k\}$ and $j \in \{\beta + 1, \ldots, n\}$.

Condition (c) deals with the remaining entries of $X_\ell(k)$, in rows $i \in \{1, 2, \ldots, k\}$ and columns $j \in \{\alpha + 1, \ldots, \beta\}$. By the definition of the assignment problem, the total number of 1-entries in the first k rows and n columns is k, α of which appear in columns $j \in \{1, 2, \ldots, \alpha\}$. Since no 1-entry can appear in columns $j \in \{\beta + 1, \ldots, n\}$, the number of the remaining 1-entries is $k - \alpha$. □

Statement 1 implies that the number of different subsets of columns which may contain 1-entries, and equivalently $|V(k)|$, depends on the selection of $(k-\alpha)$ columns out of $(\beta-\alpha)$ columns in the middle part of the solution matrix. Since $k-\alpha \leq 2\lambda$ and $\beta-\alpha \leq 4\lambda$, $|V(k)|$ can be bounded by

$$\theta = \binom{4\lambda}{2\lambda}. \tag{31}$$

Thus, the described approach reduces the assignment problem AP(2, λ) to the shortest path problem in the layered network L, where the number of layers is $n+2$ and in each layer there are no more than θ vertices. The network can be constructed in $O(|A|)$ time, where $|A| \leq \theta^2(n+1)$. An optimal solution can be found by dynamic programming in $O(|A|) = O(n)$ time as well.

Consider now the d-dimensional problem AP(d, λ) for fixed $d \geq 2$ and fixed λ. Statement 1 can be generalized by replacing 2λ by $\xi = d(d-1)\lambda$, so that

$$\begin{aligned} \alpha &= \max\{k-\xi, 0\}, \\ \beta &= \min\{k+\xi, n\}, \\ k-\alpha &= \min\{\xi, k\}, \\ \beta-k &= \min\{\xi, n-k\}. \end{aligned}$$

Any feasible d-dimensional partial solution $X_\ell(k)$ consists of k 1-entries with $i_1 \in \{1, 2, \ldots, k\}$. For each index i_z, $2 \leq z \leq d$, α of these 1-entries have $i_z \in \{1, 2, \ldots, \alpha\}$, $2 \leq z \leq d$, and the remaining $(k-\alpha)$ 1-entries have $i_z \in \{\alpha+1, \ldots, \beta\}$. Due to this, the number of nodes $|V(k)|$ in layer k depends on the selection of $(k-\alpha)$ choices for each index i_z, $2 \leq z \leq d$, out of $(\beta-\alpha)$ possible choices. Since $k-\alpha \leq \xi$ and $\beta-\alpha \leq 2\xi$, $|V(k)|$ can be bounded by Θ^{d-1}, where Θ is the amount of possible combinations for one fixed index i_z,

$$\Theta = \binom{2\xi}{\xi}. \tag{32}$$

There are at most $\Theta^{2(d-1)}$ arcs in-between two layers, if each vertex in one layer is connected to every vertex in the next one. Therefore the total number of arcs $|A|$ is bounded by $\Theta^{2(d-1)}(n+1)$, and this defines the time complexity of solving the associated shortest path problem by dynamic programming. Observe that a tighter estimate for $|A|$ can be derived with a more careful analysis of the arc set.

The complexity estimate $\Theta^{2(d-1)}(n+1)$ with Θ defined by (32) implies that problem AP(d, λ) is solvable in $O(n)$ time if d and λ are fixed, and it is fixed-parameter tractable (FPT), with parameters d and λ.

Going back to the applications discussed in Sections 2–3 we observe that they correspond to the case of $d = m$ and $\lambda = 1$, so that the described approach implies the $O(n)$ time complexity. Since modeling the two applications as AP(d, 1) incurs sorting the input data in non-decreasing order and extending the instance (as described in Section 3 after the proof of Theorem 1), we obtain the following corollary.

Corollary 1. *The problems of scheduling satellite transmissions or synchronous open shops and problem* MEC($K_{m,n}$) *with* $m = d$ *are solvable in* $O(n \log n)$ *time if* d *is fixed, and they are fixed parameter tractable if* d *is a parameter.*

We conclude this section by analyzing the maximization version of problem AP(d, λ). In that problem incompatible partners should be modeled by $-\infty$-entries in the weight array W, in order to discourage their choice. In the following, we use the notion of an inverse Monge array W: such an array satisfies the Monge property (3) with "$\leq$" replaced by "$\geq$" [5]. Equivalently, array W is inverse Monge if and only if $-W$ is Monge.

The maximization version of problem AP(d, λ) is

(i) solvable in linear time, if d is fixed and W is an inverse nearly Monge array with incompatible partners modeled by $-\infty$;
(ii) NP-hard, if $d \geq 3$ and W is a nearly Monge array with incompatible partners modeled by $-\infty$'s.

The maximization problem of type (i) with an inverse nearly Monge weight array W is equivalent to the minimization version of problem AP(d, λ) with the nearly Monge weight array $-W$; therefore the linear-time algorithm described above is applicable.

The maximization problem of type (ii) with a nearly Monge weight array is no easier than the version of the same problem with a Monge array; the latter problem is known to be NP-hard (see [5], p. 132, or [6]). The results from [6] are also discussed in Section 7.1.

7. The corridor property for other versions of the assignment problem

The corridor property that characterizes the structure of an optimal solution and restricts the search to entries around the diagonal, also holds for other versions of the assignment problem. In the literature, a property of this type was established, for example, for the three-dimensional assignment problem with decomposable costs and for the planar 3-dimensional assignment problem. We discuss these two results in Sections 7.1–7.2. Both problems deal with a min-sum objective. An alternative version, known as the bottleneck assignment problem, deals with a min–max objective; we generalize our results for that version of the assignment problem in Section 7.3.

7.1. Axial three-dimensional assignment problem with decomposable costs

The *axial three-dimensional assignment problem with decomposable costs* (3AP-DC) is defined as problem (1) with a cost array W given by $w_{ijk} = a_i b_j c_k$, where (a_i), (b_j) and (c_k) are three non-decreasing sequences of n positive numbers. Note that W is an inverse Monge array and therefore the maximization version of problem 3AP − DC is solved by the entries on the main diagonal [6]. We now focus on the minimization version of 3AP − DC. Although W does not satisfy the Monge property and in fact 3AP − DC is NP-hard [6], still there exists an optimal solution with a corridor-like structure. As proven in [6], there exists an optimal solution $\{(i^1, j^1, k^1), \ldots, (i^n, j^n, k^n)\}$ to 3AP − DC such that every triple of indices (i^g, j^g, k^g), $1 \le g \le n$, satisfies

$$n + 2 \le i^g + j^g + k^g \le 2n + 1.$$

Below we illustrate the structure of solution matrices for the two instances of problems AP(3, 1) and 3AP − DC with $n = 15$. We consider one layer for each problem with $k^g = 5$, marking feasible positions for 1-entries by $*$. For problem AP(3, 1), those positions belong to the corridor along the main diagonal with $1 \le i \le 11$ to satisfy $|i - k| \le d\,(d-1)\,\lambda = 6$; additionally they satisfy $|i - j| \le 6$. For problem 3AP − DC positions for 1-entries form a corridor that spans along the counterdiagonal, such that $17 \le i + j + 5 \le 31$.

Solution to AP(3, 1), layer $k^g = 5$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0
2	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0
3	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0
4	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0
5	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0
6	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0
7	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0
8	0	*	*	*	*	*	*	*	*	*	*	*	*	*	0
9	0	0	*	*	*	*	*	*	*	*	*	*	*	*	*
10	0	0	0	*	*	*	*	*	*	*	*	*	*	*	*
11	0	0	0	0	*	*	*	*	*	*	*	*	*	*	*
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Solution to 3AP − DC, layer $k^g = 5$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	0	0	0	0	*	*	*	*	*
2	0	0	0	0	0	0	0	0	0	*	*	*	*	*	*
3	0	0	0	0	0	0	0	0	*	*	*	*	*	*	*
4	0	0	0	0	0	0	0	*	*	*	*	*	*	*	*
5	0	0	0	0	0	0	*	*	*	*	*	*	*	*	*
6	0	0	0	0	0	*	*	*	*	*	*	*	*	*	*
7	0	0	0	0	*	*	*	*	*	*	*	*	*	*	*
8	0	0	0	*	*	*	*	*	*	*	*	*	*	*	*
9	0	0	*	*	*	*	*	*	*	*	*	*	*	*	*
10	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*
11	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
12	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0
13	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0
14	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0
15	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0

The width of the corridor for our model AP(d, λ) is fixed for given d and λ, which makes it possible to organize the search efficiently using dynamic programming. The width of the corridor for 3AP − DC depends on n, so that dynamic programming would be of exponential time complexity. Recall that 3AP − DC is NP-hard, unlike the version of problem AP(3, λ) with a fixed λ. Still the corridor property helps in the development of successful heuristics narrowing the search towards the corridor area which contains a subset of candidate triples, reduced from n^3 to $n(n^2 - 1)/3$, see [6].

7.2. Planar 3-dimensional assignment problem with a layered Monge matrix

The planar 3-dimensional assignment problem (P3AP) is another example of a model where optimal solutions have a corridor-like structure, provided each layer of the cost array is a Monge matrix, see [7]. The width of the corridor depends on the number of layers p, $p \leq n$. Formally the *p-layer planar 3-dimensional assignment problem* (p-P3AP) with an $n \times n \times p$ cost array $C = (c_{ijk})$ is defined as follows:

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{p} c_{ijk}x_{ijk} \\
\text{s.t.} \quad & \sum_{k=1}^{p} x_{ijk} \leq 1, && 1 \leq i,j \leq n, \\
& \sum_{i=1}^{n} x_{ijk} = 1, && 1 \leq j \leq n, \quad 1 \leq k \leq p, \\
& \sum_{j=1}^{n} x_{ijk} = 1, && 1 \leq i \leq n, \quad 1 \leq k \leq p, \\
& x_{ijk} \in \{0,1\}, && 1 \leq i,j \leq n, \quad 1 \leq k \leq p.
\end{aligned} \tag{33}$$

Assuming that indices i and j define rows and columns, while index k defines layers, the task is to select np 1-entries $x_{ijk} = 1$ such that

(a) in each layer k, $1 \leq k \leq p$, there are n 1-entries, one in each row and one in each column, and
(b) among all layers no 1-entry appears more than once for the same pair of indices (i, j).

The problem p-P3AP is NP-hard for every fixed $p \geq 2$. However, for the version p-P3AP$_{\text{Monge}}$ of that problem, with 2-dimensional Monge matrices C^k ($c^k_{ij} = c_{ijk}$) in every layer k, $1 \leq k \leq p$, the corridor property holds, and the problem can be solved in FPT time with respect to parameter p by dynamic programming, see [7]. In particular it is proven in [7] that there always exists an optimal solution $\{(i^g_k, j^g_k, k) | 1 \leq g \leq n, 1 \leq k \leq p\}$ to p-P3AP$_{\text{Monge}}$, such that

$$|i^g_k - j^g_k| \leq 2p - 2, \tag{34}$$

for all $1 \leq g \leq n$ and $1 \leq k \leq p$.

It is not a coincidence that the corridor condition (34) from [7] resembles the corridor condition (19) from Section 5. As we show below, the three-dimensional problem p-P3AP$_{\text{Monge}}$ with Monge matrices C^k, $1 \leq k \leq p$, reduces to the multi-dimensional problem AP(d, λ) with a nearly Monge array W.

Theorem 3. *The problem p-P3AP$_{\text{Monge}}$ reduces to problem* AP(d, λ) *with $d = p + 1$ and $\lambda = 1$,*

$$p - \text{P3AP}_{\text{Monge}} \propto \text{AP}(p+1, 1).$$

Proof. Given an instance of p-P3AP$_{\text{Monge}}$, we define an instance of AP($p+1$, 1) by introducing variables $x_{ij_1\ldots j_p}$ and the array W with entries

$$w_{ij_1\ldots j_p} = \begin{cases} c^1_{i,j_1} + c^2_{i,j_2} + \cdots + c^p_{i,j_p}, & \text{if all indices } j_1, j_2, \ldots, j_p \text{ are different,} \\ \infty, & \text{otherwise.} \end{cases} \tag{35}$$

Here, the first index i corresponds to the row index in C, indices j_k define the column indices in layers C^k, $k = 1, 2, \ldots, p$, so that $w_{i,j_1\ldots j_p}$ is the combined cost of selecting p 1-entries with the fixed first index i as part of the solution to problem (33):

$$\begin{aligned}
x_{i,j_1,1} = x_{i,j_2,2} = \cdots = x_{i,j_p,p} = 1 \quad &\Longleftrightarrow \quad x_{i,j_1\ldots j_p} = 1, \\
\sum_{k=1}^{p} c_{i,j_k,k}x_{i,j_k,k} \quad &= \quad w_{i,j_1,\ldots,j_p}x_{i,j_1,\ldots,j_p}.
\end{aligned}$$

The ∞-entries, which arise for the duplicating j-indices, prohibit assignments of the form $x_{i,j_u,u} = x_{i,j_v,v} = 1$ if $j_u = j_v$.

Every feasible solution to p-P3AP$_{\text{Monge}}$ satisfies properties (a)–(b); thus it defines a finite solution to AP($p+1$, 1) with the same value of the objective. On the other hand, every finite solution to AP($p+1$, 1) defines a feasible solution to p-P3AP$_{\text{Monge}}$ satisfying (a)–(b), also with the same value of the objective; an ∞-solution to AP($p+1$, 1) translates into an infeasible solution to p-P3AP$_{\text{Monge}}$.

It is easy to verify that the finite values of W satisfy the Monge property, so that W is a nearly Monge matrix with at most one incompatible partner for any index $j_u = j^*_u$ in every dimension corresponding to index j_v, $v \neq u$. Moreover, there are no incompatible partners for the first index i. □

The corridor property (34) derived in [7] for problem p-P3AP$_{\text{Monge}}$ can be reformulated in terms of the associated problem AP($p+1$, 1) with variables $x_{i,j_1\dots j_p}$ as

$$|j_\ell - i| \le 2p-2 \quad \text{for } 1 \le \ell \le p.$$

Notice that the corridor property established in Section 5 for problem AP(d, λ) with $d = p+1$ and $\lambda = 1$ is weaker,

$$|j_\ell - i| \le d(d-1)\lambda = (p+1)p \quad \text{for } 1 \le \ell \le p.$$

This discrepancy happens because the finite entries of array W defined by (35) satisfy not only the multi-dimensional Monge property of the form

$$w_{s\,s_1\dots s_p} + w_{t\,t_1\dots t_p} \le w_{i\,j_1\dots j_p} + w_{i'\,j'_1\dots j'_p}, \tag{36}$$

where

$$s = \min\{i, i'\}, \qquad s_\ell = \min\{j_\ell, j'_\ell\}, \quad \ell = 1,\dots,p,$$
$$t = \max\{i, i'\}, \qquad t_\ell = \max\{j_\ell, j'_\ell\}, \quad \ell = 1,\dots,p,$$

but also the condition similar to the 2-dimensional Monge property, namely for $i \le i'$ and every $\ell = 1,\dots,p$,

$$w_{ij_1\dots j_{\ell-1}\boldsymbol{s_\ell} j_{\ell+1}\dots j_p} + w_{i'j'_1\dots j'_{\ell-1}\boldsymbol{t_\ell} j'_{\ell+1}\dots j'_p} \le w_{ij_1\dots j_{\ell-1}\boldsymbol{j_\ell} j_{\ell+1}\dots j_p} + w_{i'j'_1\dots j'_{\ell-1}\boldsymbol{j'_\ell} j'_{\ell+1}\dots j'_p}, \tag{37}$$

where the changes affect the indices in bold, the remaining indices staying the same. Note that (37) implies (36), but not the other way around. The latter can be illustrated by the example with the 3-dimensional array $W^{\max}$ defined by $w^{\max}_{ijk} = \max\{i, j, k\}$, which satisfies (36) but not (37). Thus, property (37) is stronger than property (36).

In what follows we show that the proof of Theorem 2 can be adjusted to derive a corridor width of $\xi^* = 2p\lambda$ for problem AP($p+1$, λ) with a weight array W satisfying property (37). We then use ξ^* in order to derive a corridor width of $\xi^{**} = 2p-2$ if W is defined by (35), thus matching the result from [7] for problem p-P3AP$_{\text{Monge}}$. This justifies our earlier statement that the discrepancy of the two results is due to the stronger property (37) and illustrates the capabilities of the proof technique used for Theorem 2.

To prove the corridor width $\xi^* = 2p\lambda$, first consider a typical exchange step from the proof of Theorem 2 which deals with a d-tuple $(i, j_1, \dots, j_{\ell-1}, \boldsymbol{j_\ell}, j_{\ell+1}, \dots, j_p)$ with violating $\boldsymbol{j_\ell}$. For the exchange, we select a companion d-tuple $(i', j'_1, \dots, j'_{\ell-1}, \boldsymbol{j'_\ell}, j'_{\ell+1}, \dots, j'_p)$ in order to generate two induced d-tuples without increasing the value of the objective. As shown in the proof, the corridor of width ξ contains $\xi+1$ candidates for a companion d-tuple. The value $\xi = d(d-1)\lambda$ used in Theorem 2, together with the upper bound (4) on the number of incompatible partners, guarantees that at least one of the candidates is suitable for the exchange step, so that both induced d-tuples do not have incompatible indices. Recall that the exchange step in the proof of Theorem 2 affects all indices.

Condition (37) makes it possible to define simpler exchanges in comparison with those used in the proof of Theorem 2. If a d-tuple $(i, j_1, \dots, \boldsymbol{j_\ell}, \dots, j_p)$ is of Type I($\boldsymbol{j_\ell}$), then the companion d-tuple $(i', j'_1, \dots, \boldsymbol{j'_\ell}, \dots, j'_p)$ should satisfy $i' > i$ and $\boldsymbol{j'_\ell} < \boldsymbol{j_\ell}$. In the exchange step, the d-tuples $(i, j_1, \dots, j_{\ell-1}, \boldsymbol{j_\ell}, j_{\ell+1}, \dots, j_p)$ and $(i', j'_1, \dots, j'_{\ell-1}, \boldsymbol{j'_\ell}, j'_{\ell+1}, \dots, j'_p)$ corresponding to the right-hand side of (37), are replaced by the two induced d-tuples $(i, j_1, \dots, j_{\ell-1}, \boldsymbol{j'_\ell}, j_{\ell+1}, \dots, j_p)$ and $(i', j'_1, \dots, j'_{\ell-1}, \boldsymbol{j_\ell}, j'_{\ell+1}, \dots, j'_p)$, where only two indices $\boldsymbol{j_\ell}$ and $\boldsymbol{j'_\ell}$ are exchanged. The induced d-tuples do not contain incompatible indices if $\boldsymbol{j_\ell}$ is compatible with $\{i', j'_1, \dots, j'_{\ell-1}, j'_{\ell+1}, \dots, j'_p\}$ and $\boldsymbol{j'_\ell}$ is compatible with $\{i, j_1, \dots, j_{\ell-1}, j_{\ell+1}, \dots, j_p\}$. Since there are at most λp incompatible partners for $\boldsymbol{j_\ell}$ and the first set of p indices, and at most λp incompatible partners for $\boldsymbol{j'_\ell}$ and the second set of p indices, the total number of companion d-tuples for $(i, j_1, \dots, j_{\ell-1}, \boldsymbol{j_\ell}, j_{\ell+1}, \dots, j_p)$, which can lead to incompatibilities, is bounded by $2\lambda p$. Choosing the corridor width of $\xi^* = 2\lambda p$ guarantees the existence of $\xi^* + 1 = 2\lambda p + 1$ candidates for a companion d-tuple in the desirable area and therefore the existence of at least one suitable candidate among them. It is easy to make sure that similar arguments are applicable for violations of Type II($\boldsymbol{j_\ell}$).

Now consider a weight array W defined by (35). For that type of array there are no incompatible partners for first indices i, and the arguments presented in the previous paragraph can be adjusted: there are at most $\lambda(p-1)$ incompatible partners for $\boldsymbol{j_\ell}$ and at most $\lambda(p-1)$ incompatible partners for $\boldsymbol{j'_\ell}$. This implies that instead of $\xi^* = 2p\lambda$ we can consider the corridor width $\xi^{**} = 2(p-1)\lambda$. Substituting $\lambda = 1$ which holds for (35) we get the required estimate $\xi^{**} = 2p-2$.

7.3. Bottleneck assignment problem with a bottleneck nearly Monge matrix and its generalizations

Traditional research on optimization problems with Monge matrices examines first problems with min-sum objectives and then explores the counterparts with min–max objectives. In particular, the results known for the min-sum versions of the transportation and assignment problems with Monge matrices can be transferred to the *bottleneck* versions of those problems, in which "+" is replaced by "max" in the objective function (1) and in the definition of the Monge property (3), see [5]. For the bottleneck assignment problem, the goal is to minimize $\max\{w_{i_1\dots i_d} | x_{i_1\dots i_d} = 1\}$ subject to the constraints from (1), while the cost array W satisfies the bottleneck Monge property:

$$\max\{w_{s_1 s_2\dots s_d}, w_{t_1 t_2\dots t_d}\} \le \max\{w_{i_1 i_2\dots i_d}, w_{j_1 j_2\dots j_d}\}. \tag{38}$$

Table 1
The summary of complexity results for problem $AP(d, \lambda)$.

Problem	Parameters		Complexity	Reference
AP(2, λ)	$d = 2$,	λ arbitrary	$O(n^3)$	[4,21] (2-dim. assignment with an arb. cost matrix)
AP(3, λ)	$d = 3$,	λ arbitrary	str. NP-hard	[14] and Observation 3
AP(d, 0)	d arbitrary,	$\lambda = 0$	$O(n)$	[5] (assignment problem with a Monge array)
AP(d, 1)	d arbitrary,	$\lambda = 1$	str. NP-hard	[10,27] and Observation 2
AP(d, λ)	d fixed,	λ fixed	$O(n)$	Section 6

Here $s_\ell = \min\{i_\ell, j_\ell\}$ and $t_\ell = \max\{i_\ell, j_\ell\}$ for $\ell = 1, \ldots, d$. Then an optimal solution to the bottleneck assignment is given by the same n d-tuples $(1, \ldots, 1), (2, \ldots, 2), \ldots, (n, \ldots, n)$, as for the case of the linear assignment problem.

Adapting our definition of a nearly Monge matrix to the bottleneck case, we call an array W which contains ∞-entries *bottleneck nearly Monge* if condition (38) is satisfied for all finite entries. It is easy to verify that the proof of the corridor property presented in Section 5 can be modified accordingly, without affecting the width of the corridor, so that the bottleneck assignment problem with a bottleneck nearly Monge cost array can be solved in linear time by the adapted dynamic programming approach of Section 6.

The next step in extending the applicability of our results is to consider the *algebraic assignment problem* with an underlying totally ordered commutative semigroup $(H, \oplus, \preceq)$, see [5]. This problem can be considered as a generalization of both the linear and the bottleneck assignment problems. It is known that if in the algebraic assignment problem the cost array is *algebraic Monge* (i.e., condition (3) holds with "+" replaced by "$\oplus$" and "$\leq$" replaced by "$\preceq$"), then an optimal solution is of the same diagonal shape as in the case of linear assignment and bottleneck assignment.

In the context of our paper, we consider the extension of a cost array with ∞-entries. Note that the semigroup $(H, \oplus, \preceq)$ can be naturally modified to include an ∞-element if such an element does not already exist. Consider $(H^*, \oplus^*, \preceq^*)$ given by the set $H^* := H \cup \{\infty\}$, the extension $\oplus^*$ of $\oplus$ such that $a \oplus^* \infty = \infty \oplus^* a = \infty$ for all $a \in H^*$ and the extension $\preceq^*$ of $\preceq$ such that $a \preceq^* \infty$ for all $a \in H^*$. Then $(H^*, \oplus^*, \preceq^*)$ is a totally ordered commutative semi-group. We define an *algebraic nearly Monge* array as before: it is required that the algebraic Monge property should be satisfied for finite entries only. The arguments of Sections 5–6 can be adapted accordingly; notice that the proof of the corridor property only uses the properties of $(\mathbb{R} \cup \{\infty\}, +, \leq)$ that it shares with semi-groups of the form $(H^*, \oplus^*, \preceq^*)$. Thus, the proposed methodology is applicable to the algebraic assignment problem as well.

8. Conclusions

This paper presents a complexity study of the d-dimensional assignment problem with a nearly Monge array. It serves as the underlying model to applications in satellite communication and synchronous open shop scheduling, and it also models the famous MEC problem for a complete bipartite graph. A summary of the results is presented in Table 1.

In particular we study the version of problem $AP(d, \lambda)$ where the dimension d and the number of incompatible partners λ are fixed, which are natural assumptions for applications. For that special case we prove an important structural property that guides the search for an optimal solution. It allows us to limit the consideration to entries that lie inside a corridor around the diagonal. As we discuss in Section 7, the result can be extended to more general types of the assignment problem with Monge-like matrices.

Another natural extension of the current research is related to the transportation problem $TP(2, \lambda)$ with a nearly Monge matrix incurred by incompatible supply/demands pairs, that can be considered as a generalization of $AP(2, \lambda)$. In the absence of ∞-entries, the transportation problem with a square Monge matrix is solvable by a greedy algorithm in $O(n)$ time [18]. For a general square matrix the problem can be solved in $O(n^3 \log n)$ time by Orlin's algorithm [24]. It would be interesting to see if a faster than standard algorithm can be achieved for $TP(2, \lambda)$. The following example shows that the corridor property, as presented in this paper, is not quite relevant for $TP(2, \lambda)$. Consider an instance with supplies $a_1 = n - 1, a_2 = \cdots = a_n = 1$, demands $b_1 = n - 1, b_2 = \cdots = b_n = 1$ and a Monge cost matrix W. For this instance the greedy algorithm produces an optimal diagonal solution. However, introducing one incompatible pair $(1, 1)$, or equivalently $w_{11} = \infty$, makes the previous optimal solution infeasible. For the modified problem an optimal solution is defined by the first row and the first column, apart from the top left-most entry with ∞-cost. Thus, the corridor that characterizes a possible deviation from the previous optimal solution without ∞'s, is as large as the whole matrix.

Further extensions of our study can be related to the traveling salesman problem $TSP(2, \lambda)$ with a nearly Monge matrix and at most λ forbidden partners per vertex. For a Monge matrix without ∞-entries, an optimal solution is a pyramidal tour which can be found in $O(n)$ time as shown in [25]. Clearly, if we allow an arbitrary number of infinities in the matrix, then finding a finite solution is as hard as finding a Hamiltonian circuit in an arbitrary graph, and therefore strongly NP-hard [19]. On the other hand, for Supnick matrices, which can be viewed as a subclass of nearly Monge matrices with $\lambda = 1$, the TSP is always solved by the pyramidal tour $(1, 3, 5, \ldots, n, \ldots, 6, 4, 2)$, see [32]. An interesting question related to $TSP(2, \lambda)$ with infinities is how far an optimal solution may deviate from the pyramidal tour if λ is a fixed parameter. Pyramidal tours with step-backs, as introduced in [12], might be a good starting point for that study.

Design of approximation algorithms for problems with nearly Monge matrices is another important research direction. With respect to $AP(d, \lambda)$ with arbitrary d, the closest problem is MEC, in which the cost matrix is of type (5). A range

of approximation algorithms for the latter problem is proposed in [8,10,13,22,23]. These ideas might be helpful to find approximation results for AP(d, λ), which is a generalization of MEC, as the cost values may be arbitrary, unrelated to the max-formula (5).

Acknowledgments

The work of S. Knust and S. Waldherr was supported by the Deutsche Forschungsgemeinschaft, KN 512/7-1. The work of N.V. Shakhlevich was supported by the EPSRC grant EP/K041274/1. We want to thank Samuel Wilson for many useful discussions.

We are very grateful for the comments of two anonymous reviewers who helped us to improve the presentation of the paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.dam.2016.04.019.

References

[1] A. Aggarwal, J.K. Park, Notes on searching in multidimensional monotone arrays, in: Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science, 1988, pp. 497–512.
[2] W.W. Bein, P. Brucker, J.K. Park, P.K. Pathak, A Monge property for the d-dimensional transportation problem, Discrete Appl. Math. 58 (1995) 97–109.
[3] R.E. Burkard, Monge properties, discrete convexity and applications, European J. Oper. Res. 176 (2007) 1–14.
[4] R.E. Burkard, M. Dell'Amico, S. Martello, Assignment Problems, SIAM, Philadelphia, USA, 2009.
[5] R.E. Burkard, B. Klinz, R. Rudolf, Perspectives of Monge properties in optimization, Discrete Appl. Math. 70 (1996) 95–161.
[6] R.E. Burkard, R. Rudolf, G.J. Woeginger, Three-dimensional axial assignment problems with decomposable cost coefficients, Discrete Appl. Math. 65 (1996) 123–139.
[7] A. Ćustić, B. Klinz, G.J. Woeginger, Planar 3-dimensional assignment problems with Monge-like cost arrays. 2014, E-print, arXiv:1405.5210.
[8] D. de Werra, M. Demange, B. Escoffier, J. Monnot, V.T. Paschos, Weighted coloring on planar, bipartite and split graphs: Complexity and approximation, Discrete Appl. Math. 157 (2009) 819–832.
[9] V.G. Deineko, R. Rudolf, G.J. Woeginger, On the recognition of permuted Supnick and incomplete Monge matrices, Acta Inform. 33 (1996) 559–569.
[10] M. Demange, D. de Werra, J. Monnot, V.T. Paschos, Weighted node coloring: when stable sets are expensive, Lecture Notes in Comput. Sci. 2573 (2002) 114–125.
[11] M. Demange, B. Escoffier, G. Lucarelli, I. Milis, J. Monnot, V.T. Paschos, D. de Werra, Weighted edge coloring, in: V.T. Paschos (Ed.), Combinatorial Optimization and Theoretical Computer Science, ISTE, London, 2008.
[12] H. Enomoto, Y. Oda, K. Ota, Pyramidal tours with step-backs and the asymmetric traveling salesman problem, Discrete Appl. Math. 87 (1998) 57–65.
[13] B. Escoffier, J. Monnot, V.T. Pashos, Weighted coloring: further complexity and approximability results, Inform. Process. Lett. 97 (2006) 98–103.
[14] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Co., San Francisco, USA, 1979.
[15] I.S. Gopal, C.K. Wong, Minimising the number of switchings in an SS/TDMA system, IEEE Trans. Commun. 33 (1985) 497–501.
[16] D.S. Hochbaum, R. Shamir, Minimizing the number of tardy job unit under release time constraints, Discrete Appl. Math. 28 (1990) 45–57.
[17] D.S. Hochbaum, R. Shamir, Strongly polynomial algorithms for the high multiplicity scheduling problem, Oper. Res. 39 (1991) 648–653.
[18] A.J. Hoffman, On simple linear programming problems, in: ed. V. Klee, Convexity: Proceedings of the Seventh Symposium in Pure Mathematics of the AMS, in: Proceedings of Symposia in Pure Mathematics, vol. 7, 1963, pp. 317–327.
[19] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, New York, 1972, pp. 85–104.
[20] A. Kesselman, K. Kogan, Nonpreemptive scheduling of optical switches, IEEE Trans. Commun. 55 (2007) 1212–1219.
[21] H.W. Kuhn, The Hungarian method for the assignment problem, Naval Res. Logist. Q. 2 (1955) 83–97.
[22] G. Lucarelli, I. Millis, Improved approximation algorithms for the max edge-coloring problem, Inform. Process. Lett. 111 (2011) 819–823.
[23] G. Lucarelli, I. Millis, V.T. Paschos, On the max-weight edge coloring problem, J. Comb. Optim. 20 (2010) 429–442.
[24] J.B. Orlin, A faster strongly polynomial minimum cost flow algorithm, Oper. Res. 41 (1993) 338–350.
[25] J.K. Park, A special case of the n-vertex traveling-salesman problem that can be solved in $O(n)$ time, Inform. Process. Lett. 40 (1991) 247–254.
[26] M. Queyranne, F. Spieksma, F. Tardella, A general class of greedily solvable linear programs, Math. Oper. Res. 23 (1998) 892–908.
[27] F. Rendl, On the complexity of decomposing matrices arising in satellite communication, Oper. Res. Lett. 4 (1985) 5–8.
[28] C.C. Ribeiro, M. Minoux, M.C. Penna, An optimal column-generation-with-ranking algorithm for very large scale set partitioning problems in traffic assignment, European J. Oper. Res. 41 (1989) 232–239.
[29] R. Rudolf, Recognition of d-dimensional Monge arrays, Discrete Appl. Math. 52 (1994) 71–82.
[30] R. Shamir, A fast algorithm for constructing Monge sequences in transportation problems with forbidden arcs, Discrete Math. 114 (1993) 435–444.
[31] B. Soylu, Ö Kirca, M. Azizoğ lu, Flow shop-sequencing problem with synchronous transfers and makespan minimization, Int. J. Prod. Res. 45 (2007) 3311–3331.
[32] F. Supnick, Extreme Hamiltonian lines, Ann. of Math. 66 (1957) 179–201.
[33] S. Waldherr, S. Knust, Complexity results for flow shop problems with synchronous movement, European J. Oper. Res. 242 (2015) 34–44.