

# Approximability and exact resolution of the Multidimensional Binary Vector Assignment problem

Marin Bougeret, Guillaume Duvillié, Rodolphe Giroudeau

► **To cite this version:**

Marin Bougeret, Guillaume Duvillié, Rodolphe Giroudeau. Approximability and exact resolution of the Multidimensional Binary Vector Assignment problem. [Research Report] Lirmm; Montpellier II. 2016. <lirmm-01310648>

**HAL Id: lirmm-01310648**

**<https://hal-lirmm.ccsd.cnrs.fr/lirmm-01310648>**

Submitted on 2 May 2016

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



# Approximability and exact resolution of the Multidimensional Binary Vector Assignment problem

M. Bougeret<sup>1</sup>, G. Duvalié<sup>1</sup>, R. Giroudeau<sup>1</sup>

LIRMM, Université Montpellier 2, France

{marin.bougeret,guillerme.duvalie,rodolphe.giroudeau}@lirmm.fr

**Abstract.** In this paper we consider the multidimensional binary vector assignment problem. An input of this problem is defined by  $m$  disjoint sets  $V^1, V^2, \dots, V^m$ , each composed of  $n$  binary vectors of size  $p$ . An output is a set of  $n$  disjoint  $m$ -tuples of vectors, where each  $m$ -tuple is obtained by picking one vector from each set  $V^i$ . To each  $m$ -tuple we associate a  $p$  dimensional vector by applying the bit-wise AND operation on the  $m$  vectors of the tuple. The objective is to minimize the total number of zeros in these  $n$  vectors. We denote this problem by  $\min \sum 0$ , and the restriction of this problem where every vector has at most  $c$  zeros by  $(\min \sum 0)_{\#0 \leq c}$ .  $(\min \sum 0)_{\#0 \leq 2}$  was only known to be **APX**-complete, even for  $m = 3$  [5]. We show that, assuming the unique games conjecture, it is **NP**-hard to  $(n - \varepsilon)$ -approximate  $(\min \sum 0)_{\#0 \leq 1}$  for any fixed  $n$  and  $\varepsilon$ . This result is tight as any solution is a  $n$ -approximation. We also prove without assuming UGC that  $(\min \sum 0)_{\#0 \leq 1}$  is **APX**-complete even for  $n = 2$ , and we provide an example of  $n - f(n, m)$ -approximation algorithm for  $\min \sum 0$ . Finally, we show that  $(\min \sum 0)_{\#0 \leq 1}$  is polynomial-time solvable for fixed  $m$  (which cannot be extended to  $(\min \sum 0)_{\#0 \leq 2}$  according to [5]).

## 1 Introduction

### 1.1 Problem definition

In this paper we consider the multidimensional binary vector assignment problem denoted by  $\min \sum 0$ . An input of this problem (see Figure 1) is described by  $m$  disjoint sets  $V^1, \dots, V^m$ , each set  $V^i$  containing  $n$  binary  $p$ -dimensional vectors. For any  $j \in [n]^1$ , and any  $i \in [m]$ , the  $j^{\text{th}}$  vector of set  $V^i$  is denoted  $v_j^i$ , and for any  $k \in [p]$ , the  $k^{\text{th}}$  coordinate of  $v_j^i$  is denoted  $v_j^i[k]$ .

The output of the problem consists in a set  $S$  of  $n$  disjoint stacks. A stack  $s = (v_1^s, \dots, v_m^s)$  is an  $m$ -tuple of vectors such that  $v_i^s \in V^i$ , for any  $i \in [m]$ . Two stacks  $s_1$  and  $s_2$  are disjoint if and only if no vector belongs to  $s_1$  and  $s_2$ .

We now introduce the operator  $\wedge$  which assigns to a pair of vectors  $(u, v)$  the vector given by  $u \wedge v = (u[1] \wedge v[1], u[2] \wedge v[2], \dots, u[p] \wedge v[p])$ . We associate to each stack  $s$  a unique vector given by  $v_s = \bigwedge_{i \in [m]} v_i^s$ .

---

<sup>1</sup> Note that  $[n]$  stands for  $\{1, 2, \dots, n\}$ .

The cost of a vector  $v$  is defined as the number of zeros in it. More formally if  $v$  is  $p$ -dimensional,  $c(v) = p - \sum_{k \in [p]} v[k]$ . We extend this definition to a set of stacks  $S = \{s_1, \dots, s_n\}$  as follows :  $c(S) = \sum_{s \in S} c(v_s)$ .

The objective is then to find a set  $S$  of  $n$  disjoint stacks minimizing the total number of zeros. This leads us to the following definition of the problem:

---

**Optimization Problem 1**  $\min \sum 0$

**Input**  $m$  sets of  $n$   $p$ -dimensional binary vectors.

**Output** A set  $S$  of  $n$  disjoint stacks minimizing  $c(S)$ .

---

Throughout this paper, we denote  $(\min \sum 0)_{\#0 \leq c}$  the restriction of  $\min \sum 0$  where the number of zeros per vector is upper bounded by  $c$ .

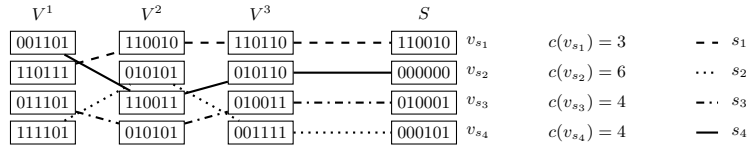


Fig. 1: Example of  $\min \sum 0$  instance with  $m = 3, n = 4, p = 6$  and of a feasible solution  $S$  of cost  $c(S) = 17$ .

## 1.2 Related work

The dual version of the problem called  $\max \sum 1$  (where the objective is to maximize the total number of 1 in the created stacks) has been introduced by Reda et al. in [8] as the “yield maximization problem in Wafer-to-Wafer 3-D Integration technology”. They prove the **NP**-completeness of  $\max \sum 1$  and provide heuristics without approximation guarantee. In [6] we proved that, even for  $n = 2$ , for any  $\varepsilon > 0$ ,  $\max \sum 1$  is  $\mathcal{O}(m^{1-\varepsilon})$  and  $\mathcal{O}(p^{1-\varepsilon})$  inapproximable unless **P** = **NP**. We also provide an ILP formulation proving that  $\max \sum 1$  (and thus  $\min \sum 0$ ) is **FPT**<sup>2</sup> when parameterized by  $p$ .

We introduced  $\min \sum 0$  in [4] where we provide in particular  $\frac{4}{3}$ -approximation algorithm for  $m = 3$ . In [5], authors focus on a generalization of  $\min \sum 0$ , called MULTI DIMENSIONAL VECTOR ASSIGNMENT, where vectors are not necessary binary vectors. They extend the approximation algorithm of [4] to get a  $f(m)$ -approximation algorithm for arbitrary  $m$ . They also prove the **APX**-completeness of the  $(\min \sum 0)_{\#0 \leq 2}$  for  $m = 3$ . This result was the only known inapproximability result for  $\min \sum 0$ .

## 1.3 Contribution

In section 2 we study the approximability of  $\min \sum 0$ . Our main result in this section is to prove that assuming UGC, it is **NP**-hard to  $(n - \varepsilon)$ -approximate

<sup>2</sup> *i.e.* admits an algorithm in  $f(p)poly(|I|)$  for an arbitrary function  $f$ .

$(\min \sum 0)_{\#0 \leq 1}$  (and thus  $\min \sum 0$ ) for any fixed  $n \geq 2$ ,  $\forall \varepsilon > 0$ . This result is tight as any solution is a  $n$ -approximation.

Notice that this improves the only existing negative result for  $\min \sum 0$ , which was the **APX**-hardness of [5] (implying only no-**PTAS**).

We also show how this reduction can be used to obtain the **APX**-hardness for  $(\min \sum 0)_{\#0 \leq 1}$  for  $n = 2$  unless  $\mathbf{P} = \mathbf{NP}$ , which is weaker negative result, but does not require UGC. We then give an example  $n - f(n, m)$  approximation algorithm for the general problem  $\min \sum 0$ .

In section 3, we consider the exact resolution of  $\min \sum 0$  (and  $\max \sum 1$ ). We only focus on what we will call sparse instances, *i.e.* instances of  $(\min \sum 0)_{\#0 \leq 1}$ . Indeed, recall that authors of [5] show that  $(\min \sum 0)_{\#0 \leq 2}$  is **APX**-complete even for  $m = 3$ , implying that  $(\min \sum 0)_{\#0 \leq 2}$  cannot be polynomial-time solvable for fixed  $m$  unless  $\mathbf{P} = \mathbf{NP}$ . Thus, it was natural to ask if  $(\min \sum 0)_{\#0 \leq 1}$  was polynomial-time solvable for fixed  $m$ . Section 3 is devoted to answer positively to this question. Notice that the question of determining if  $(\min \sum 0)_{\#0 \leq 1}$  is **FPT** when parameterized by  $m$  remains open. Due to space constraints, results marked with a  $\star$  are proved in the appendix.

## 2 Approximability of $\min \sum 0$

Let us first recall definitions of reductions we use in this paper.

### 2.1 Definitions

**L-reduction** The  $L$ -reduction has been introduced by Papadimitriou et al. in [7] as follows:

**Definition 1.** Let  $\Pi_1$  and  $\Pi_2$  be two optimization problems with objective functions  $m_1$  and  $m_2$ . Let  $f$  be a polynomial-time computable function that given any instance  $x$  of  $\Pi_1$  associates an instance  $f(x)$  of  $\Pi_2$ . Let  $g$  be another polynomial-time computable function that given any instance  $x$  of  $\Pi_1$ , and feasible solution  $S$  of  $f(x)$ , associates a feasible solution  $g(x, S)$  of  $\Pi_1$ . If  $f$  and  $g$  verify the two following conditions:

1.  $\exists \alpha$  such that  $Opt(f(x)) \leq \alpha Opt(x)$
2.  $\exists \beta$  such that for each solution  $S$  of  $\Pi_2$ ,  $|Opt(x) - m_1(g(x, S))| \leq \beta |Opt(f(x)) - m_2(S)|$

then  $(f, g)$  is an  $L$ -reduction.

In following,  $\Pi_1$   $L$ -reduces to  $\Pi_2$  is noted  $\Pi_1 <_L \Pi_2$ .

**Gap reduction** We briefly recall the definition of such a reduction, as presented in [2] by Ausiello et al.

**Definition 2.** Let  $\Pi_{dec}$  be a decision problem and  $\Pi_{opt}$  a minimization problem. Let  $f$  be a polynomial-time computable function that given any instance  $x$  of  $\Pi_{dec}$  associates an instance  $f(x)$  of  $\Pi_{opt}$ . If there exists two function  $a$  and  $r$  such that:

1.  $x$  is a YES-instance  $\Rightarrow \text{Opt}(f(x)) \leq a(x)$
2.  $x$  is a NO-instance  $\Rightarrow \text{Opt}(f(x)) \geq r(x)a(x)$

then  $f$  is a  $r(x)$ -Gap reduction.

## 2.2 Inapproximability results for $(\min \sum 0)_{\#0 \leq 1}$

From now we suppose that  $\forall k \in [p], \exists i, \exists j$  such that  $v_j^i[k] = 0$ . In other words, for any solution  $S$  and  $\forall k$ , there exists a stack  $s$  such that  $v_s[k] = 0$ . Otherwise, we simply remove such a coordinate from every vector of every set, and decrease  $p$  by one. Since this coordinate would be set to 1 in all the stacks of all solutions, such a preprocessing preserves approximation ratios and exact results.

In a first time, we define the following polynomial-time computable function  $f$  which associates an instance of  $(\min \sum 0)_{\#0 \leq 1}$  to any  $k$ -uniform hypergraph, *i.e.* an hypergraph  $G = (U, E)$  such that every hyperedges of  $E$  contains exactly  $k$  distinct elements of  $U$ .

**Definition of  $f$**  We consider a  $k$ -uniform hypergraph  $G = (U, E)$ . We call  $f$  the polynomial-time computable function that creates an instance of  $(\min \sum 0)_{\#0 \leq 1}$  from a  $G$  as follows.

1. We set  $m = |E|$ ,  $n = k$  and  $p = |U|$ .
2. For each hyperedge  $e = \{u_1, u_2, \dots, u_k\} \in E$ , we create the set  $V^e$  containing  $k$  vectors  $\{v_j^e, j \in [k]\}$ , where for all  $j \in [k]$ ,  $v_j^e[u_j] = 0$  and  $v_j^e[l] = 1$  for  $l \neq u_j$ . We say that a vector  $v$  **represents**  $u \in U$  iff  $v[u] = 0$  and  $v[l \neq u] = 1$  (and thus vector  $v_j^e$  represents  $u_j$ ).

An example of this construction is given in Figure 2.

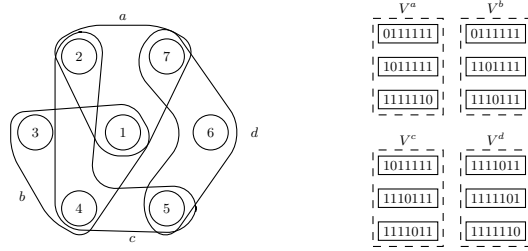


Fig. 2: Illustration of the reduction from an hypergraph  $G = (U = \{1, 2, 3, 4, 5, 6, 7\}, E = \{\{1, 2, 7\}, \{1, 3, 4\}, \{2, 4, 5\}, \{5, 6, 7\}\})$  to an instance  $(\min \sum 0)_{\#0 \leq 1}$

**Negative results assuming UGC** We consider the following problem. Notice that what we call a vertex cover in a  $k$ -regular hypergraph  $G = (U, E)$  is a set  $U' \subseteq U$  such that for any hyperedge  $e \in E$ ,  $U' \cap e \neq \emptyset$ .

---

**Decision Problem 1** ALMOST  $Ek$  VERTEX COVER

**Input** We are given an integer  $k \geq 2$ , two arbitrary positive constants  $\varepsilon$  and  $\delta$  and a  $k$ -uniform hypergraph  $G = (U, E)$ .

**Output** Distinguish between the following cases:

**YES Case** there exist  $k$  disjoint subsets  $U^1, U^2, \dots, U^k \subseteq U$ , satisfying  $|U^i| \geq \frac{1-\varepsilon}{k}|U|$  and such that every hyperedge contains at most one vertex from each  $U^i$ .

**NO Case** every vertex cover has size at least  $(1 - \delta)|U|$ .

---

It is shown in [3] that, assuming UGC, this problem is **NP**-complete.

**Theorem 1.** For any fixed  $n \geq 2$ , for any constants  $\varepsilon, \delta > 0$ , there exists a  $\frac{n-n\delta}{1+n\varepsilon}$ -Gap reduction from ALMOST  $Ek$  VERTEX COVER to  $(\min \sum 0)_{\#0 \leq 1}$ . Consequently, under UGC, for any fixed  $n$   $(\min \sum 0)_{\#0 \leq 1}$  is **NP**-hard to approximate within a factor  $(n - \varepsilon')$  for any  $\varepsilon' > 0$ .

*Proof.* We consider an instance  $I$  of ALMOST  $Ek$  VERTEX COVER defined by two positive constants  $\delta$  and  $\varepsilon$ , an integer  $k$  and a  $k$ -regular hypergraph  $G = (U, E)$ .

We use the function  $f$  previously defined to construct an instance  $f(I)$  of  $\min \sum 0$ . Let us now prove that if  $I$  is a positive instance,  $f(I)$  admits a solution  $S$  of cost  $c(S) < (1 + n\varepsilon)|U|$ , and otherwise any solution  $S$  of  $f(I)$  has cost  $c(S) \geq n(1 - \delta)|U|$ .

**NO Case** Let  $S$  be a solution of  $f(I)$ . Let us first remark that for any stack  $s \in S$ , the set  $\{k : v_s[k] = 0\}$  defines a vertex cover in  $G$ . Indeed,  $s$  contains exactly one vector per set, and thus by construction  $s$  selects one vertex per hyperedge in  $G$ . Remark also that the cost of  $s$  is equal to the size of the corresponding vertex cover.

Now, suppose that  $I$  is a negative instance. Hence each vertex cover has a size at least equal to  $(1 - \delta)|U|$ , and any solution  $S$  of  $f(I)$ , composed of exactly  $n$  stacks, verifies  $c(S) \geq n(1 - \delta)|U|$ .

**YES Case** If  $I$  is a positive instance, there exists  $k$  disjoint sets  $U^1, U^2, \dots, U^k \subseteq U$  such that  $\forall i = 1, \dots, k, |U^i| \geq \frac{1-\varepsilon}{k}|U|$  and such that every hyperedge contains at most one vertex from each  $U^i$ .

We introduce the subset  $X = U \setminus \bigcup_{i=1}^k U^i$ . By definition  $\{U^1, U^2, \dots, U^k, X\}$  is a partition of  $U$  and  $|X| \leq \varepsilon|U|$ . Furthermore,  $U^i \cup X$  is a vertex cover  $\forall i = 1, \dots, k$ . Indeed, each hyperedge  $e \in E$  that contains no vertex of  $U^i$ , contains at least one vertex of  $X$  since  $e$  contains  $k$  vertices.

We now construct a solution  $S$  of  $f(I)$ . Our objective is to construct stacks  $\{s_i\}$  such that for any  $i$ , the zeros of  $s_i$  are included in  $U^i \cup X$  (i.e.  $\{l : v_{s_i}[l] = 0\} \subseteq U^i \cup X$ ). For each  $e = \{u_1, \dots, u_k\} \in E$ , we show how to assign exactly one vector of  $V^e$  to each stack  $s_1, \dots, s_k$ . For all  $i \in [k]$ , if  $v_j^e$  represents a vertex  $u$  with  $u \in U^i$ , then we assign  $v_j^e$  to  $s_i$ . W.l.o.g., let

$S'_e = \{s_1, \dots, s_{k'}\}$  (for  $k' \leq k$ ) be the set of stacks that received a vertex during this process. Notice that as every hyperedge contains at most one vertex from each  $U^i$ , we only assigned one vector to each stack of  $S'_e$ . After this, every unassigned vector  $v \in V^e$  represents a vertex of  $X$  (otherwise, such a vector  $v$  would belong to a set  $U^i$ ,  $i \in k'$ , a contradiction). We assign arbitrarily these vectors to the remaining stacks that are not in  $S'_e$ . As by construction  $\forall i \in [k]$ ,  $v_s i$  contains only vectors representing vertices from  $U^i \cup X$ , we get  $c(s_i) \leq |U^i| + |X|$ .

Thus, we obtain a feasible solution  $S$  of cost  $c(S) = \sum_{i=1}^k c(s_i) \leq k|X| + \sum_{i=1}^k |U^i|$ . As by definition we have  $|X| + \sum_{i=1}^k |U^i| = |U|$ , it follows that  $c(S) \leq |U| + (k-1)\varepsilon|U|$  and since  $k = n$ ,  $c(S) < |U|(1+n\varepsilon)$ .

If we define  $a(n) = (1+n\varepsilon)|U|$  and  $r(n) = \frac{n(1-\delta)}{(1+n\varepsilon)}$ , the previous reduction is a  $r(n)$ -Gap reduction. Furthermore,  $\lim_{\delta, \varepsilon \rightarrow 0} r(n) = n$ , thus it is **NP**-hard to approximate  $(\min \sum 0)_{\#0 \leq 1}$  within a ratio  $(n - \varepsilon')$  for any  $\varepsilon' > 0$ .  $\square$

Notice that, as a function of  $n$ , this inapproximability result is optimal. Indeed, we observe that any feasible solution  $S$  is an  $n$ -approximation as, for any instance  $I$  of  $\min \sum 0^3$ ,  $Opt(I) \geq p$  and for any solution  $S$ ,  $c(S) \leq pn$ .

**Negative results without assuming UGC** Let us now study the negative results we can get when only assuming  $\mathbf{P} \neq \mathbf{NP}$ . Our objective is to prove that  $(\min \sum 0)_{\#0 \leq 1}$  is **APX**-hard, even for  $n = 2$ . To do so, we present a reduction from **ODD CYCLE TRANSVERSAL**, which is defined as follows. Given an input graph  $G = (U, E)$ , the objective is to find an odd cycle transversal of minimum size, *i.e.* a subset  $T \subseteq U$  of minimum size such that  $G[U \setminus T]$  is bipartite.

For any integer  $\gamma \geq 2$ , we denote  $\mathcal{G}_\gamma$  the class of graphs  $G = (U, E)$  such that any optimal odd cycle transversal  $T$  has size  $|T| \geq \frac{|U|}{\gamma}$ . Given  $\mathcal{G}$  a class of graphs, we denote  $OCT_{\mathcal{G}}$  the **ODD CYCLE TRANSVERSAL** problem restricted to  $\mathcal{G}$ .

**Lemma 1.** *For any constant  $\gamma \geq 2$ , there exists an L-reduction from  $OCT_{\mathcal{G}_\gamma}$  to  $(\min \sum 0)_{\#0 \leq 1}$  with  $n = 2$ .*

*Proof.* Let us consider an integer  $\gamma$ , an instance  $I$  of  $OCT_{\mathcal{G}_\gamma}$ , defined by a graph  $G = (V, E)$  such that  $G \in \mathcal{G}_\gamma$ . W.l.o.g., we can consider that  $G$  contains no isolated vertex.

Remark that any graph can be seen as a 2-uniform hypergraph. Thus, we use the function  $f$  previously defined to construct an instance  $f(I)$  of  $(\min \sum 0)_{\#0 \leq 1}$  such that  $n = 2$ . Since,  $G$  contains no isolated vertex,  $f(I)$  contains no position  $k$  such that  $\forall i \in [m]$ ,  $\forall j \in [n]$ ,  $v_j^i[k] = 1$ .

Let us now prove that  $I$  admits an odd cycle transversal of size  $t$  if and only if  $f(I)$  admits a solution of cost  $p + t$ .

<sup>3</sup> Recall that we assume  $\forall k \in [p], \exists i, \exists j$  such that  $v_j^i[k] = 0$

$\Leftarrow$  We consider an instance  $f(I)$  of  $(\min \sum 0)_{\#0 \leq 1}$  with  $n = 2$  admitting a solution  $S = \{s_A, s_B\}$  with cost  $c(S) = p + t$ . Let us specify a function  $g$  which produces from  $S$  a solution  $T = g(I, S)$  of  $OCT_{\mathcal{G}_\gamma}$ , *i.e.* a set of vertices of  $U$  such that  $G[U \setminus T]$  is bipartite.

We define  $T = \{u \in U : v_{s_A}[u] = v_{s_B}[u] = 0\}$ , the set of coordinates equal to zero in both  $s_A$  and  $s_B$ . We also define  $A = \{u \in V : v_{s_A}[u] = 0 \text{ and } v_{s_B}[u] = 1\}$  (resp.  $B = \{u \in V : v_{s_B}[u] = 0 \text{ and } v_{s_A}[u] = 1\}$ ), the set of coordinates set to zero only in  $s_A$  (resp.  $s_B$ ). Notice that  $\{T, A, B\}$  is a partition of  $U$ .

Remark that  $A$  and  $B$  are independent sets. Indeed, suppose that  $\exists \{u, v\} \in E$  such that  $u, v \in A$ . As  $\{u, v\} \in E$  there exists a set  $V^{(u,v)}$  containing a vector that represents  $u$  and another vector that represents  $v$ , and thus these vectors are assigned to different stacks. This leads to a contradiction. It follows that  $G[U \setminus T]$  is bipartite and  $T$  is an odd cycle transversal.

Since  $c(S) = |A| + |B| + 2|T| = p + |T| = p + t$ , we get  $|T| = t$ .

$\Rightarrow$  We consider an instance  $I$  of  $OCT_{\mathcal{G}_\gamma}$  and a solution  $T$  of size  $t$ . We now construct a solution  $S = \{s_A, s_B\}$  of  $f(I)$  from  $T$ .

By definition,  $G[U \setminus T]$  is a bipartite graph, thus the vertices in  $U \setminus T$  may be split into two disjoint independent sets  $A$  and  $B$ . For each edge  $e \in E$ , the following cases can occur:

- if  $\exists u \in e$  such that  $u \in A$ , then the vector corresponding to  $u$  is assigned to  $s_A$ , and the vector corresponding to  $e \setminus \{u\}$  is assigned to  $s_B$  (and the same rule holds by exchanging  $A$  and  $B$ )
- otherwise,  $u$  and  $v \in T$ , and we assign arbitrarily  $v_u^e$  to  $s_A$  and the other to  $s_B$ .

We claim that the stacks  $s_A$  and  $s_B$  describe a feasible solution  $S$  of cost at most  $p + t$ .

Since, for each set, only one vector is assigned to  $s_A$  and the other to  $s_B$ , the two stacks  $s_A$  and  $s_B$  are disjoint and contain exactly  $m$  vectors.  $S$  is therefore a feasible solution.

Remark that  $v_{s_A}$  (resp.  $v_{s_B}$ ) contains only vectors  $v$  such that  $v[k] = 0 \implies k \in A \cup T$  (resp.  $k \in B \cup T$ ), and thus  $c(v_A) \leq |A| + |T|$  (resp.  $c(v_B) \leq |B| + |T|$ ). Hence  $c(S) \leq |A| + |B| + 2|T| = p + t$ .

Let us now prove that this reduction is an  $L$ -reduction.

1. By definition, any instance  $I$  of  $OCT_{\mathcal{G}_\gamma}$  verifies  $|Opt(I)| \geq |U|/\gamma$ . Thus,

$$Opt(f(I)) \leq |U| + Opt(I) \leq (\gamma + 1)Opt(I)$$

2. We consider an arbitrary instance  $I$  of  $OCT_{\mathcal{G}_\gamma}$ ,  $f(I)$  the corresponding instance of  $(\min \sum 0)_{\#0 \leq 1}$ ,  $S$  a solution of  $f(I)$  and  $T = g(I, S)$  the corresponding solution of  $I$ .

We proved  $|T| - Opt(I) = c(S) - |U| - (Opt(f(I)) - |U|) = c(S) - Opt(f(I))$ .

Therefore, we get an  $L$ -reduction for  $\alpha = \gamma + 1$  and  $\beta = 1$ . □



**Lemma 2.** *There exist a constant  $\gamma$  and  $\mathcal{G} \subset \mathcal{G}_\gamma$  such that  $OCT_{\mathcal{G}}$  is **APX-hard**.*

*Proof.* We present an  $L$ -reduction from VC-3, the vertex cover problem in graph with maximum degree 3, to  $OCT_{\mathcal{G}_{VC}}$  for an appropriate  $\mathcal{G}_{VC}$ . VC-3 is known to be **APX-complete** [1].

Given an instance  $G = (U, E)$  of VC-3, we construct an instance  $f(G) = (U', E')$  as follows:

1. For each  $(u, v) \in E$ , create a vertex  $z_{u,v}$ . These  $z$ -vertices form the set  $Z$ .
2.  $U' = U \cup Z$ .
3.  $E' = E \cup \{(u, z_{u,v}), (v, z_{u,v}) : (u, v) \in E\}$ . In other words, for each  $(u, v) \in E$ , we create the triangle  $\{u, v, z_{u,v}\}$ .

Let us prove that  $G = (U, E)$  admits a solution  $VC$  of size  $|VC| = t$  if and only if  $f(G)$  admits a solution  $T$  of size  $|T| = t$ .

$\Rightarrow$  Consider a vertex cover  $VC$  of size  $|VC| = t$ , for each  $u \in VC$ , we add the vertex  $u'$  to  $T$ . By definition,  $VC$  covers all the edges of  $G$  and then all its (odd) cycles. Furthermore, it also covers all the created triangles in  $f(G)$  since each of these cycles contains exactly one edge in common with  $f(G)[U' \setminus Z]$ . Thus  $T$  is an odd cycle transversal and  $|T| = |VC|$ .

$\Leftarrow$  Let us construct a function  $g$  that, given any solution  $T$  of  $f(G)$ , computes a solution  $VC = g(G, T)$  of  $G$ . Notice first that we can suppose that  $T$  contains no  $z$ -vertex. Otherwise every triangle  $\{u, v, z_{u,v}\}$  covered by a  $z_{u,v} \in T$ , can instead be covered by either  $u$  or  $v$  without increasing the size of  $T$ . Thus, we set  $VC = T$ .

By definition of an odd cycle transversal,  $T$  covers all the odd cycles of  $f(G)$  and especially the created triangles. Thus, the triangle  $\{u, v, z_{u,v}\}$  corresponding to any edge  $\{u, v\} \in E$  is covered by  $VC$ . As  $VC \cap Z = \emptyset$ ,  $VC$  is a vertex cover of  $G$ .

The previous reduction is an  $L$ -reduction for  $\alpha = \beta = 1$ . Let us call  $\mathcal{G}_{VC}$  the class of graph generated in this reduction. The previous reduction shows that  $OCT_{\mathcal{G}_{VC}}$  is **APX-hard**. It remains to check that  $\mathcal{G}_{VC} \subseteq \mathcal{G}_\gamma$  for a constant  $\gamma$ .

Remark that VC-3 is only defined on 3-regular graphs, it implies that for any instance  $G = (U, E)$  of VC-3,  $Opt(G) \geq \frac{|U|}{3}$ . As  $|U'| = |U| + |E| \leq \frac{5|U|}{2}$ , it follows that  $Opt(f(G)) = Opt(G) \geq \frac{|U|}{3} \geq \frac{2|U'|}{15}$ . Hence,  $\mathcal{G}_{VC} \subset \mathcal{G}_\gamma$  with  $\gamma = \frac{15}{2}$ .  $\square$

The following result is now immediate.

**Theorem 2.**  $(\min \sum_{\#0 \leq 1} 0)$  is **APX-hard**, even for  $n = 2$ .

### 2.3 Approximation algorithm for $\min \sum 0$

Let us now show an example of algorithm achieving a  $n - f(n, m)$  ratio. Notice that the  $(n - \epsilon)$  inapproximability result holds for fixed  $n$  and  $\#0 = 1$ , while the following algorithm is polynomial-time computable when  $n$  is part of the input and  $\#0$  is arbitrary.

**Proposition 1.** *There is a polynomial-time  $n - \frac{n-1}{n\rho(n,m)}$  approximation algorithm for  $\min \sum 0$ , where  $\rho(n, m) > 1$  is the approximation ratio for independent set in graphs that are the union of  $m$  complete  $n$ -partite graphs.*

*Proof.* Let  $I$  be an instance of  $\min \sum 0$ . Let us now consider an optimal solution  $S^* = \{s_1^*, \dots, s_n^*\}$  of  $I$ . For any  $i \in [n]$ , let  $Z_i^* = \{l \in [p] : v_{s_i^*}[l] = 0 \text{ and } v_{s_t^*}[l] = 1, \forall t \neq i\}$  be the set of coordinates equal to zero only in stack  $s_i^*$ . Let  $\Delta = \sum_{i=1}^n |Z_i^*|$ . Notice that we have  $c(S^*) \geq \Delta + 2(p - \Delta)$ , as for any coordinate  $l$  outside  $\bigcup_i Z_i^*$ , there are at least two stacks with a zero at coordinate  $l$ . W.l.o.g., let us suppose that  $Z_1^*$  is the largest set among  $\{Z_i^*\}$ , implying  $|Z_1^*| \geq \frac{\Delta}{n}$ .

Given a subset  $Z \subset [p]$ , we will construct a solution  $S = \{s_1, \dots, s_n\}$  such that for any  $l \in Z$ ,  $v_{s_1}[l] = 0$ , and for any  $i \neq 1$ ,  $v_{s_i}[l] = 1$ . Informally, the zero at coordinates  $Z$  will appear only in  $s_1$ , which behaves as a "trash" stack. The cost of such a solution is  $c(S) \leq c(s_1) + \sum_{i=2}^n c(s_i) \leq p + (n - 1)(p - |Z|)$ . Our objective is now to compute such a set  $Z$ , and to lower bound  $|Z|$  according to  $|Z_1^*|$ .

Let us now define how we compute  $Z$ . Let  $P = \{l \in [p] : \forall i \in [m], |\{j : v_j^i[l] = 0\}| \leq 1\}$  be the subset of coordinates that are never nullified in two different vectors of the same set. We will construct a simple undirected graph  $G = (P, E)$ , and thus it remains to define  $E$ . For vector  $v_j^i$ , let  $Z_j^i = Z(v_j^i) \cap P$ , where  $Z(v) \subseteq [p]$  denotes the set of null coordinates of vector  $v$ . For any  $i \in [m]$ , we add to  $G$  the edges of the complete  $n$ -partite graph  $G^i = (\{Z_1^i \times \dots \times Z_n^i\})$  (i.e. for any  $j_1, j_2, v_1 \in Z_{j_1}^i, v_2 \in Z_{j_2}^i$ , we add edge  $\{v_1, v_2\}$  to  $G$ ). This concludes the description of  $G$ , which can be seen as the union of  $m$  complete  $n$ -partite graphs.

Let us now see the link between independent set in  $G$  and our problem. Let us first see why  $Z_1^*$  is a independent set in  $G$ . Recall that by definition of  $Z_1^*$ , for any  $l \in Z_1^*$ ,  $v_{s_1^*}[l] = 0$ , but  $v_{s_j^*}[l] = 1, j \geq 2$ . Thus, it is immediate that  $Z_1^* \subseteq P$ . Moreover, assume by contradiction that there exists an edge in  $G$  between two vertices  $l_1$  and  $l_2$  of  $Z_1^*$ . This implies that there exists  $i \in [m], j_1$  and  $j_2 \neq j_1$  such that  $v_{j_1}^i[l_1] = 0$  and  $v_{j_2}^i[l_2] = 0$ . As by definition of  $Z_1^*$  we must have  $v_{s_j^*}[l_1] = 1$  and  $v_{s_j^*}[l_2] = 1$  for  $j \geq 2$ , this implies that  $s_1^*$  must contains both  $v_{j_1}^i$  and  $v_{j_2}^i$ , a contradiction. Thus, we get  $OPT(G) \geq |Z_1^*|$ , where  $OPT(G)$  is the size of a maximum independent set in  $G$ .

Now, let us check that for any independent set  $Z \subseteq P$  in  $G$ , we can construct a solution  $S = \{s_1, \dots, s_n\}$  such that for any  $l \in Z$ ,  $v_{s_1}[l] = 0$ , and for any  $i \neq 1$ ,  $v_{s_i}[l] = 1$ . To construct such a solution, we have to prove that we can add in  $s_1$  all the vectors  $v$  such that  $\exists l \in Z$  such that  $v[l] = 0$ . However, this last

statement is clearly true as for any  $i \in [m]$ , there is at most one vector  $v_j^i$  with  $Z(v_j^i) \subseteq Z$ .

Thus, any  $\rho(n, m)$  approximation algorithm gives us a set  $Z$  with  $|Z| \geq \frac{|Z_1^*|}{\rho(n, m)} \geq \frac{\Delta}{n\rho(n, m)}$ , and we get a ratio of  $\frac{p+(n-1)(p-\frac{\Delta}{n\rho(n, m)})}{2p-\Delta} \leq n - \frac{n-1}{n\rho(n, m)}$  for  $\Delta = p$ .  $\square$

*Remark 1.* We can get, for example,  $\rho(n, m) = mn^{m-1}$  using the following algorithm. For any  $i \in [m]$ , let  $G^i = (A_1^i, \dots, A_n^i)$  be the  $i$ -th complete  $n$ -partite graph. W.l.o.g., suppose that  $A_1^1$  is the largest set among  $\{A_j^i\}$ . Notice that  $|A_1^1| \geq \frac{OPT}{m}$ . The algorithm starts by setting  $S_1 = A_1^1$  ( $S_1$  may not be an independent set). Then, for any  $i$  from 2 to  $m$ , the algorithm set  $S_i = S_{i-1} \setminus (\cup_{j \neq j_0} A_j^i)$ , where  $j_0 = \arg \max_j \{|S_{i-1} \cap A_j^i|\}$ . Thus, for any  $i$  we have  $|S_i| \geq \frac{|S_{i-1}|}{n}$ , and  $S_i$  is an independent set when considering only edges from  $\cup_{l=1}^i G^l$ . Finally, we get an independent set of  $G$  of size  $|S_m| \geq \frac{|S_1|}{n^{m-1}} \geq \frac{OPT}{mn^{m-1}}$ .

### 3 Exact resolution of sparse instances

The section is devoted to the exact resolution of  $\min \sum 0$  for sparse instances where each vector has at most one zero ( $\#0 \leq 1$ ). As we have seen in Section 2,  $(\min \sum 0)_{\#0 \leq 1}$  remains **NP**-hard (even for  $n = 2$ ). Thus it is natural to ask if  $(\min \sum 0)_{\#0 \leq 1}$  is polynomial-time solvable for fixed  $m$  (for general  $n$ ). This section is devoted to answer positively to this question. Notice that we cannot extend this result to a more general notion of sparsity as  $(\min \sum 0)_{\#0 \leq 2}$  is **APX**-complete for  $m = 3$  [5]. However, the question if  $(\min \sum 0)_{\#0 \leq 1}$  is fixed parameter tractable when parameterized by  $m$  is left open.

We first need some definitions, and refer the reader to Figure 3 where an example is depicted.

#### Definition 3.

- For any  $l \in [p], i \in [m]$ , we define  $B^{(l, i)} = \{v_j^i : v_j^i[l] = 0\}$  to be the set of vectors of set  $i$  that have their (unique) zero at position  $l$ . For the sake of homogeneous notation, we define  $B^{(p+1, i)} = \{v_j^i : v_j^i \text{ is a } 1 \text{ vector}\}$ . Notice that the  $B^{(l, i)}$  form a partition of all the vectors of the input, and thus an input of  $(\min \sum 0)_{\#0 \leq 1}$  is completely characterized by the  $B^{(l, i)}$ .
- For any  $l \in [p+1]$ , the **block**  $B^l = \cup_{i \in [m]} B^{(l, i)}$ .

Informally, the idea to solve  $(\min \sum 0)_{\#0 \leq 1}$  in polynomial time for fixed  $m$  is to parse the input block after block using a dynamic programming algorithm. When arriving at block  $B^l$  we only need to remember for each  $c \subseteq [m]$  the number  $x_c$  of “partial stacks” that have only one vector for each  $V^i, i \in c$ . Indeed, we do not need to remember what is “inside” these partial stacks as all the remaining vectors from  $B^{l'}, l' \geq l$  cannot “match” (*i.e.* have their zero in the same position) the vectors in these partial stacks.

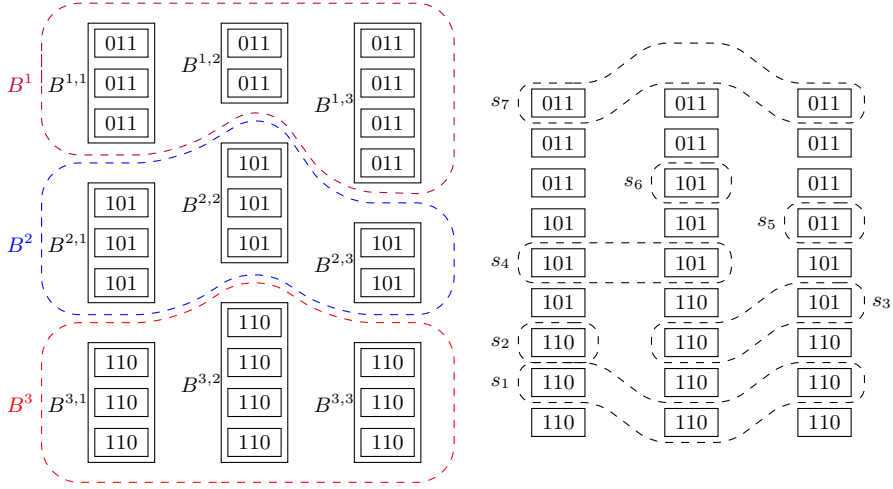


Fig. 3: Left: instance  $I$  of  $(\min \sum 0)_{\#0 \leq 1}$  partitioned into blocks. Right: A profile  $P = \{x_{\{\emptyset\}} = 2, x_{\{1\}} = 1, x_{\{2\}} = 1, x_{\{3\}} = 1, x_{\{1,2\}} = 1, x_{\{1,3\}} = 1, x_{\{2,3\}} = 1, x_{\{1,2,3\}} = 1\}$  encoding a set  $S$  of partial stacks of  $I$  containing two empty stacks. The support of  $s_7$  is  $\text{sup}(s_7) = \{1, 3\}$  and has cost  $c(s_7) = 1$ .

#### Definition 4.

- A **partial stack**  $s = \{v_{i_1}^s, \dots, v_{i_k}^s\}$  of  $I$  is such that  $\{i_x \in [m], x \in [k]\}$  are pairwise disjoint, and for any  $x \in [k]$ ,  $v_{i_x}^s \in V^{i_x}$ . The **support** of a partial stack  $s$  is  $\text{sup}(s) = \{i_x, x \in [k]\}$ . Notice that a stack  $s$  (i.e. non partial) has  $\text{sup}(s) = [m]$ .
- The cost is extended in the natural way: the cost of a partial stack  $c(s) = c(\bigwedge_{x \in [k]} v_{i_x}^s)$  is the number of zeros of the bitwise AND of the vectors of  $s$ .

We define the notion of profile as follows:

**Definition 5.** A **profile**  $P = \{x_c, c \subseteq [m]\}$  is a set of  $2^m$  positive integers such that  $\sum_{c \subseteq [m]} x_c = n$ .

In the following, a profile will be used to encode a set  $S$  of  $n$  partial stacks by keeping a record of their support. In other words,  $x_c, c \subseteq [m]$  will denote the number of partial stacks in  $S$  of support  $c$ . This leads us to introduce the notion of reachable profile as follows:

**Definition 6.** Given two profiles  $P = \{x_c : c \subseteq [m]\}$  and  $P' = \{x'_c : c' \subseteq [m]\}$  and a set  $S = \{s_1, \dots, s_n\}$  of  $n$  partial stacks,  $P'$  is said **reachable** from  $P$  through  $S$  iff there exist  $n$  couples  $(s_1, c_1), (s_2, c_2), \dots, (s_n, c_n)$  such that:

- For each couple  $(s, c)$ ,  $\text{sup}(s) \cap c = \emptyset$ .
- For each  $c \subseteq [m]$ ,  $|\{(s_j, c_j) : c_j = c, j = 1, \dots, n\}| = x_c$ . Intuitively, the configuration  $c$  appears in exactly  $x_c$  couples.

- For each  $c' \subseteq [m]$ ,  $|\{(s'_j, c'_j) : \text{sup}(s'_j) \cup c'_j = c', j = 1, \dots, n\}| = x'_{c'}$ . Intuitively, there exist exactly  $x'_{c'}$  couples that, when associated, create a partial of profile  $c'$ .

Given two profiles  $P$  and  $P'$ ,  $P$  is said reachable from  $P'$ , if there exists a set  $S$  of  $n$  partial stacks such that  $P'$  is reachable from  $P$  through  $S$ .

Intuitively, a profile  $P'$  is reachable from  $P$  through  $S$  if every partial stack of the set encoded by  $P$  can be assigned to a unique partial stack from  $S$  to obtain a set of new partial stacks encoded by  $P'$ .

Remark that, given a set of partial stacks  $S$  only their profile is used to determine whether a profile is reachable or not. An example of a reachable profile is given on Figure 4.

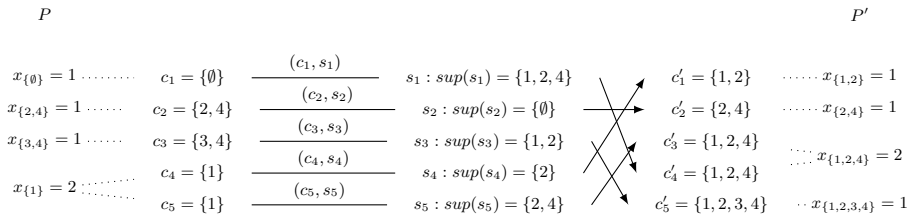


Fig. 4: Example of a profile  $P' = \{x_{\{1,2\}} = 1, x_{\{2,4\}} = 1, x_{\{1,2,4\}} = 2, x_{\{1,2,3,4\}} = 1\}$  that is reachable from  $P = \{x_{\{\emptyset\}} = 1, x_{\{1\}} = 2, x_{\{2,4\}} = 1, x_{\{3,4\}} = 1\}$  reachable through  $S = \{s_1 : \text{sup}(s_1) = \{1, 2, 4\}, s_2 : \text{sup}(s_2) = \{\emptyset\}, s_3 : \text{sup}(s_3) = \{1, 2\}, s_4 : \text{sup}(s_4) = \{2\}, s_5 : \text{sup}(s_5) = \{2, 4\}\}$ .

We introduce now the following problem  $II$ . We then show that this problem can be used to solve  $(\min \sum 0)_{\#0 \leq 1}$  problem, and we present a dynamic programming algorithm that solves  $II$  in polynomial time when  $m$  is fixed.

---

### Optimization Problem 2 $II$

**Input**  $(l, P)$  with  $l \in [p + 1]$ ,  $P$  a profile.

**Output** A set of  $n$  partial stacks  $S = \{s_1, s_2, \dots, s_n\}$  such that  $S$  is a partition of  $\mathcal{B} = \bigcup_{l' \geq l} B^{l'}$  and for every  $c \subseteq [m]$ ,  $|\{s \in S | \text{sup}(s) = [m] \setminus c\}| = x_c$  and such that  $c(S) = \sum_{j=1}^n c(s_j)$  is minimum.

---

Remark that an instance  $I$  of  $(\min \sum 0)_{\#0 \leq 1}$  can be solved optimally by solving optimally the instance  $I' = (1, P = \{x_{\{\emptyset\}} = n, x_c = 0, \forall c \neq \{\emptyset\}\})$  of  $II$ . The optimal solution of  $I'$  is indeed a set of  $n$  partial disjoint stacks of support  $[m]$  of minimum cost.

We are now ready to define the following dynamic programming algorithm that solves any instance  $(l, P)$  of  $II$  by parsing the instance block after block and branching for each of these blocks on every reachable profile.

---

**Function** MinSumZeroDP( $l, P$ )
 

---

**if**  $k == p + 1$  **then**  
     **return** 0;  
**return**  $\min(c(S') + \text{MinSumZeroDP}(l + 1, P'))$ , with  $P'$  reachable from  $P$   
 through  $S'$ , where  $S'$  partition of  $B^l$ ;

---

Note that this dynamic programming assumes the existence of a procedure that enumerates *efficiently* all the profiles  $P'$  that are reachable from  $P$ . The existence of such a procedure will be shown thereafter.

**Lemma 3.** *For any instance of  $\Pi$  ( $l, P$ ),  $\text{MinSumZeroDP}(l, P) = \text{Opt}(l, P)$ .*

*Proof.* Lemma 3 is true as in a given block  $l$ , the algorithm tries every reachable profile, and the zeros of vectors in blocks  $\mathcal{B} = \bigcup_{l' < l} B^{l'}$  cannot be matched with those of vectors in block  $\mathcal{B}' = \bigcup_{l' \geq l} B^{l'}$ . This is the reason why the support of the already created partial stacks (stored in profile  $P$ ) is sufficient to keep a record of what have been done (the positions of the zeros in the partial stacks corresponding to  $P$  is not relevant).  $\square$

Let us focus now on the procedure in charge of the enumeration of the reachable profile. A first and intuitive way to perform this operation is by guessing, for all  $c, c' \subseteq [m]$ ,  $y_{c,c'}$  the number of partial stacks in configuration  $c$  that will be turned into configuration  $c'$  with vectors of current block  $B^l$ . For each such guess it is possible to greedily verify that each  $y_{c,c'}$  can be satisfied with the vectors of the current block. As each of the  $y_{c,c'}$  can take values from 0 to  $n$  and  $c$  and  $c'$  can be both enumerated in  $\mathcal{O}^*(n^{2^m})$ , the previous algorithm runs in  $\mathcal{O}^*(n^{2^{2m}})$ .

This complexity can be improved as follows. The idea is to enumerate every possible profile  $P'$  and to verify using another dynamic programming algorithm if such a  $P'$  is reachable from  $P$ . We define  $\text{Aux}_{P'}(P, X)$ , that verifies if  $P'$  is reachable from  $P$  by using all vectors of  $X$ . If  $X = \emptyset$ , then the algorithm returns whether  $P$  is equal to  $P'$  or not. Otherwise, we consider the first vector  $v$  of  $X$  (we fix any arbitrary order) for which a branching is done on every possible assignment of  $v$ . More formally, the algorithm returns  $\bigvee_{c \subseteq [m], x_c > 0, c \cap \text{sup}(v) = \emptyset} \text{Aux}_{P'}(P_2 = \{x'_l\}, X \setminus \{v\})$ , where  $x'_l = x_l - 1$  if  $l = c$ ,  $x'_l = x_l + 1$  if  $l = c \cup \text{sup}(v)$ , and  $x'_l = x_l$  otherwise.

Using  $\text{Aux}$  in  $\text{MinSumZeroDP}$ , we get the following theorem.

**Theorem 3.**  $(\min \sum_0)_{\#0 \leq 1}$  can be solved in  $\mathcal{O}^*(n^{2^{m+2}})$ .

We compute the overall complexity as follows: for each of the  $pn^{2^m}$  possible values of the parameters of  $\text{MinSumZeroDP}$ , the algorithm tries the  $n^{2^m}$  profiles  $P'$ , and run for each one  $\text{Aux}_{P'}$  in  $\mathcal{O}^*(n^{2^m} nm)$  (the first parameter of  $\text{Aux}$  can take  $n^{2^m}$  values, and the second  $nm$  as we just encode how many vectors left in  $X$ ).

## References

1. P. Alimonti and V. Kann. Some APX-completeness results for cubic graphs. *Theoretical Computer Science*, 237(1-2):123–134, 2000.
2. G. Ausiello and V. T. Paschos. Reductions, completeness and the hardness of approximability. *European Journal of Operational Research*, 172(3):719–739, 2006.
3. N. Bansal and S. Khot. Inapproximability of hypergraph vertex cover and applications to scheduling problems. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 250–261, 2010.
4. T. Dokka, M. Bougeret, V. Boudet, R. Giroudeau, and F. C. Spieksma. Approximation algorithms for the wafer to wafer integration problem. In *Approximation and Online Algorithms (WAOA)*, pages 286–297. Springer, 2013.
5. T. Dokka, Y. Crama, and F. C. Spieksma. Multi-dimensional vector assignment problems. *Discrete Optimization*, 14:111–125, 2014.
6. G. Duvallié, M. Bougeret, V. Boudet, T. Dokka, and R. Giroudeau. On the complexity of wafer-to-wafer integration. In *International Conference on Algorithms and Complexity (CIAC)*, pages 208–220, 2015.
7. C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 229–234. ACM, 1988.
8. S. Reda, G. Smith, and L. Smith. Maximizing the functional yield of wafer-to-wafer 3-d integration. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 17(9):1357–1362, 2009.