Approximability and exact resolution of the multidimensional binary vector assignment problem

Marin Bougeret\textsuperscript{1} · Guillerme Duvillié\textsuperscript{1,2} · Rodolphe Giroudeau\textsuperscript{1}

Published online: 29 March 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract In this paper we consider the multidimensional binary vector assignment problem. An input of this problem is defined by \(m\) disjoint multisets \(V^1, V^2, \ldots, V^m\), 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 multiset \(V^i\). 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 \(\text{APX}\)-hard, even for \(m = 3\). We show that, assuming the unique games conjecture, it is \(\text{NP}\)-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 \(\text{APX}\)-hard even for \(n = 2\). Finally, we show that \((\min \sum 0)_{\#0 \leq 1}\) is polynomial-time solvable for fixed \(m\) (which cannot be extended to \((\min \sum 0)_{\#0 \leq 2}\)).

This work is supported by the Interuniversity Attraction Poles Programme P7/36 \(\approx \text{COMEX}\) \(\approx\) initiated by the Belgian Science Policy Office.

\textsuperscript{1} LIRMM, 161 rue Ada, 34095 Montpellier, France
\textsuperscript{2} Université libre de Bruxelles, Avenue F.D. Roosevelt 50, 1050 Brussels, Belgium
Keywords Approximation algorithm · UGC · Inapproximability · Dynamic programming

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 Fig. 1) is described by \( m \) multisets \( V^1, \ldots, V^m \), each multiset \( V^i \) containing \( n \) binary \( p \)-dimensional vectors. If we denote \( [n] \) the set \( \{1, 2, \ldots, n\} \), then for any \( j \in [n] \), and any \( i \in [m] \), the \( j^{th} \) vector of multiset \( V^i \) is denoted \( v^i_j \), and for any \( k \in [p] \), the \( k^{th} \) coordinate of \( v^i_j \) is denoted \( v^i_j[k] \).

The objective of this problem is to create a set \( S \) of \( n \) stacks. A stack \( s = (v^s_1, \ldots, v^s_m) \) is an \( m - tuple \) of vectors such that \( v^s_i \in V^i \), for any \( i \in [m] \). Furthermore, \( S \) has to be such that every vector of the input appears in exactly one created stack.

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

The cost of a vector \( v \) is defined as the number of zeros it contains. 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 \( S = \{s_1, \ldots, s_n\} \) as follows : \( c(S) = \sum_{i \in S} c(v^s_i) \).

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** \( m \) multisets of \( n \) \( p \)-dimensional binary vectors.

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

This problem is motivated by an application in integrated circuit manufacturing in semiconductor industry and is also known as Wafer-to-Wafer Integration problem. The objective is to manufacture 3D-microprocessors by superimposing wafers instead of dies (see Reda et al. 2009 for more details about this application). A wafer can be seen as a string of bad dies (0) and good dies (1). Integrating two wafers corresponds to superimposing the two corresponding strings. In this operation, a position in the merged string is only ‘good’ when the two corresponding dies are good, otherwise it is ‘bad’.

Throughout this paper, we consider an extremal case of this problem denoted as \( \left( \min \sum 0 \right)_{\#0 \leq c} \). The latter is the restriction of \( \min \sum 0 \) where the number of zeros per vector in the input is upper bounded by \( c \).
1.2 Related work

The dual version of the problem called max $\sum 1$ (where the objective is to maximize the total number of ones in the created stacks) has been introduced by Reda et al. in Reda et al. (2009) 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 Duvillié et al. (2015) we proved that, even for $n = 2$, for any $\varepsilon > 0$, max $\sum 1$ is $O(m^{1-\varepsilon})$ and $O(p^{1-\varepsilon})$ inapproximable unless $P = NP$. We also provide an ILP formulation proving that max $\sum 1$ (and thus min $\sum 0$) is FPT when parameterized by $p$.

We introduced min $\sum 0$ in Dokka et al. (2013) where we provide in particular $\frac{4}{3}$-approximation algorithm for $m = 3$. In Dokka et al. (2014), 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 Dokka et al. (2013) to get a $f(m)$-approximation algorithm for arbitrary $m$, where $f(m) = \frac{1}{2}(m + 1) - \frac{1}{4}(\ln(m - 1))$. They also prove the APX-completeness of $(\min \sum 0)_{\#0\leq2}$ for $m = 3$. This result was the only known inapproximability result for min $\sum 0$.

1.3 Contribution

In Sect. 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 $(\min \sum 0)_{\#0\leq1}$ (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 Dokka et al. (2014) (implying only no-PTAS).

We also show how this reduction can be used to obtain the APX-hardness for $(\min \sum 0)_{\#0\leq1}$ for $n = 2$, which is weaker negative result, but does not require UGC.

In Sect. 3, we consider the exact resolution of min $\sum 0$. We focus on sparse instances, i.e., instances of $(\min \sum 0)_{\#0\leq1}$. Indeed, recall that authors of Dokka et al. (2014) show that $(\min \sum 0)_{\#0\leq2}$ is APX-complete when $m = 3$, implying that $(\min \sum 0)_{\#0\leq2}$ cannot be polynomial-time solvable for fixed $m$ unless $P = NP$.

---

1 i.e. admits an algorithm in $f(p)poly(|I|)$ for an arbitrary function $f$. 

\[2\] Springer
Thus, it is natural to ask if \((\min \sum 0)_{\#0 \leq 1}\) is 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 \(\text{FPT}\) when parameterized by \(m\) remains open.

### 1.4 Assumptions

Throughout this paper, we make the following assumptions:

**Hypothesis 1** We suppose that for all set \(V^i\), there always exist \(j \in [n], k \in [p]\) such that \(v^i_j[k] = 0\).

In other words, we consider instances that do not contain a set consisting only in perfect vectors, i.e., vectors with no coordinate set to zero. Given an instance of \(\min \sum 0\), if it contains at least one of those perfect sets, the latter can be removed without altering the cost of the optimal solution. Indeed, perfect vectors can be seen as neutral element for the bitwise AND operation.

Based on the same remark, we can define a similar hypothesis at the coordinate level.

**Hypothesis 2** We suppose that for all set \(V^i\) and for every coordinate \(k \in [p]\), there always exists \(j \in [n]\) such that \(v^i_j[k] = 1\).

Let us consider an instance that violates Hypothesis 2 and thus such that given a set \(V^i\) and a coordinate \(k \in [p]\), \(v^i_j[k] = 0\), \(\forall j \in [n]\). In such a configuration, any solution \(S\) satisfies \(v_s[k] = 0\) for every stack \(s \in S\). The coordinate \(k\) can thus be removed from every vector of the instance. Note that this can lead to an alteration of the solution cost depending on the objective function, and thus legitimizes questions about approximability results for these problems\(^2\). However, these problems are minimization problems. Hence any approximation algorithm for instances of \(\min \sum 0\) satisfying 2 would provide an approximation algorithm with at least the same ratio for instances of \(\min \sum 0\) containing one or more zero coordinates. It is sufficient to delete the zero-coordinates, applying the algorithm on the created instance, and adding back the deleted coordinates. When adding back the zero-coordinates, the cost of the returned solution and of the optimal one are increased by the same amount and the performance ratio of the returned solution is improved.

A last hypothesis can be done at the coordinate level.

**Hypothesis 3** We suppose that for every coordinate \(k \in [p]\), there always exist \(i \in [m]\) and \(j \in [n]\) such that \(v^i_j[k] = 0\).

Roughly speaking, for any coordinate \(k \in [p]\), there exists at least one vector in the whole instance that has the \(k^{th}\) coordinate set to zero. Otherwise, any solution \(S\) would satisfy \(v_s[k] = 1\) for every stack \(s \in S\). Such coordinates can then be removed from the instance. We use same argument, but for maximization problems, to show that such an operation is safe from the approximability point of view.

\(^2\) Incriminated problems are \(\min \min 0\) and \(\min \sum 0\).
Based on this Hypothesis, we can easily show that any solution is an $n$-approximation. Indeed, thanks to Hypothesis $3$, we can write that, given any instance $I$ of $\min \sum 0$, $c(\text{Opt}(I)) \geq p$. Furthermore, any solution $S$ satisfies $c(S) \leq np$.

2 Approximability of $\min \sum 0$

2.1 Prolegomena

In this paper, we consider Gap, Strict and $L$-reductions as respectively defined in Vazirani (2001), Crescenzi (1997) and Papadimitriou and Yannakakis (1988). Let us recall their definition.

**Definition 1** (Gap reduction Vazirani (2001)) Let $\Pi_d$ a decision problem and $\Pi_o$ an NPO minimization problem with objective function $m$. A reduction $(f, g)$ from $\Pi_d$ to $\Pi_o$ is said to be an $r$-Gap reduction if there exist two computable functions $a$ and $r$ such that, for every instance $I$ of $\Pi_d$:

1. $I$ is a positive instance implies that the optimal solution $\text{Opt}(I)$ of $f(I)$ has cost $m(f(I), \text{Opt}(I)) \leq a(I)$,
2. $I$ is a negative instance implies that the optimal solution $\text{Opt}(I)$ of $f(I)$ has cost $m(f(I), \text{Opt}(I)) \geq r(I).a(I)$.

**Property 1** If there exists an $r$-Gap-Reduction from an NP-hard decision problem $\Pi_d$ to an NPO problem $\Pi_o$, thus, for any $\varepsilon > 0$, $\Pi_o$ does not admit a $(r - \varepsilon)$-approximation algorithm unless $P = NP$.

The definition of the Strict-reduction is given as follows:

**Definition 2** (Strict-reduction Crescenzi (1997)) Given two NPO minimization problems $\Pi_1, \Pi_2$ and a reduction $(f, g)$ from $\Pi_1$ to $\Pi_2$, if for any instance $I$ of $\Pi_1$ and for any solution $x$ in the constructed instance $f(I)$ of $\Pi_2$:

$$\frac{m(I, g(I, x))}{m(I, \text{Opt}(I))} \leq \frac{m(f(I), x)}{m(f(I), \text{Opt}(f(I)))}$$

A Strict-reduction satisfies the following property:

**Property 2** A Strict-reduction preserves the membership in PTAS and APX.

At last, the definition of the $L$-reduction is the following one:

**Definition 3** (L-reduction Papadimitriou and Yannakakis (1988)) Given two NPO problems $\Pi_1, \Pi_2$ and a reduction $(f, g)$ from $\Pi_1$ to $\Pi_2$, if there exist $\alpha, \beta \in \mathbb{R}^+,*$ such that, for any instance $I$ of $\Pi_1$, any optimal solution $\text{Opt}(I)$ of $I$ and for any optimal solution $\text{Opt}(f(I))$ and any solution $x_2$ of the associated instance $f(I)$ of $\Pi_2$, the following holds:

$$m_2(f(I), \text{Opt}(f(I))) \leq \alpha m_1(I, \text{Opt}(I))$$ (1)
\[ \|m_1(I, Opt(I)) - m_1(I, g(I, x))\| \leq \beta|m_2(f(I), Opt(f(I))) - m_2(f(I), x)| \]

(2)

A particularity of this reduction is that the conclusions that can be done from the existence of such a polynomial-time reduction differ in function of the type of the initial problem, as explained in Property 3.

**Property 3** An \(L\)-reduction from an **NPO** problem \(\Pi_1\) to another **NPO** problem \(\Pi_2\) preserves membership in:

1. **FPTAS** if \(\Pi_1\) is a maximization problem,
2. **FPTAS** and **APX** if \(\Pi_1\) is a minimization problem.

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

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 \(U\).

#### 2.2.1 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 \(G\) as follows.

1. We set \(m = |E|, n = k\) and \(p = |U|\).
2. For each hyperedge \(e = \{u_1, u_2, \ldots, u_k\} \in E\), we create the set \(V^e\) containing \(k\) vectors \(\{v^e_j, j \in [k]\}\), where for all \(j \in [k]\), \(v^e_j[u_j] = 0\) and \(v^e_j[l] = 1\) 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 \(v^e_j\) represents \(u_j\)).

An example of this construction is given in Fig. 2.

#### 2.2.2 Negative results assuming UGC

In the following, a vertex cover in a \(k\)-uniform hypergraph \(G = (U, E)\) is defined as a set \(U' \subseteq U\) such that for any hyperedge \(e \in E\), \(U' \cap e \neq \emptyset\).

We consider the Decision Problem 1, denoted as \((\varepsilon, \delta)\)-**Almost \(Ek\** VERTEX COVER. Informally, given an integer \(k \geq 2\) and a \(k\)-uniform hypergraph that verifies exactly one of the two following properties:

1. it only admits very large vertex covers,
2. it is almost \(k\) partite,

the problem aims at determining which property is verified by the hypergraph. The latter is formally defined as follows:
**Decision Problem 1** $(\varepsilon, \delta)$-Almost $E_k$ Vertex Cover

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

**Output** Distinguish between the following cases:

YES Case there exist $k$ disjoint subsets $U_1, U_2, \ldots, U_k \subseteq U$, 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 Bansal and Khot (2010) that, assuming UGC, this problem is NP-complete for any $\varepsilon, \delta > 0$. Such a negative results leads to the following Theorem:

**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 $(\varepsilon, \delta)$-Almost $E_k$ 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 $(\varepsilon, \delta)$-Almost $E_k$ Vertex Cover defined by an integer $k$ and a $k$-uniform 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 Let us suppose $I$ to be a positive instance. In this case, there exist $k$ disjoint sets $U^1, U^2, \ldots, U^k \subseteq U$ such that $\forall i = 1, \ldots, k$, $|U^i| \geq \frac{1-\epsilon}{k}|U|$ 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, the set of sets $\{U^1, U^2, \ldots, U^k, X\}$ is a partition of $U$ and $X \leq \epsilon|U|$. Furthermore, $U^i \cup X$ is a vertex cover $\forall i = 1, \ldots, k$. 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 $e = \{u_1, \ldots, u_k\} \in E$, we show how to assign exactly one vector of $V^e$ to each stack $s_1, \ldots, s_k$. For all $i \in [k]$, if $v^e_i$ represents a vertex $u$ with $u \in U^i$, then we assign $v^e_i$ to $s_i$. W.l.o.g., let $S'_e = \{s_1, \ldots, s_{k'}\}$ (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 $U^i$, $i \leq k'$, a contradiction). We assign arbitrarily these vectors to the remaining uncomplete stacks that are not in $S'_e$.

As by construction $\forall i \in [k]$, $s_i$ 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 $c(S) \leq |U|+(k-1)\epsilon|U|$ and since $k = n, c(S) < |U|(1+n\epsilon)$.

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

Notice that, as a function of $n$, this inapproximability result is optimal. Indeed, as previously stated, any feasible solution $S$ is an $n$-approximation.

2.2.3 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 $\mathbf{APX}$-hard, even for $n = 2$. To do so, we present a reduction from $\text{ODD CYCLE TRANSVERSAL}$, which is defined as follows.

$\text{Springer}$
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 $G$ a class of graphs, we denote $OCT_G$ the Odd Cycle Transversal problem restricted to $G$.

**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 $\forall i \in [m]$, $\forall j \in [n]$, $v^f_{i,k} = 1$.

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 $p + t$.

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

We define $T = \{u \in U : v^A_{s_A}[u] = v^B_{s_B}[u] = 0\}$, the set of coordinates equal to zero in both $s_A$ and $s_B$. We define $A = \{u \in V : v^A_{s_A}[u] = 0 \text{ and } v^B_{s_B}[u] = 1\}$ (resp. $B = \{u \in V : v^A_{s_A}[u] = 0 \text{ and } v^B_{s_B}[u] = 1\}$), 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 $|T| = t$.

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

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 $v^e_u$ 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 $v_{sA}$ (resp. $v_{sB}$) contains only vectors $v$ such that $v[k] = 0 \implies k \in A \cup T$ (resp. $k \in B \cup T$), and thus $c(v_{sA}) \leq |A| + |T|$ (resp. $c(v_{sB}) \leq |B| + |T|$). 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,

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

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

We proved that $|T| = c(S) - p$ and that $|Opt(I)| = c(Opt(f(I))) - p$. Since $p = |U|$, we can write $|T| - |Opt(I)| = c(S) - |U| - (c(Opt(f(I))) - |U|) = c(S) - c(Opt(f(I)))$.

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 Strict-reduction from VC-3, the VERTEX COVER problem on graph with maximum degree 3, to $OCT_{\mathcal{G}_{VC}}$, where $\mathcal{G}_{VC}$ is the class of graphs generated by the reduction. VC-3 is known to be APX-complete (Alimonti and Kann 2000).

Let us define the functions $f$ and $g$ depicted in Definition 2. 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 hits all its (odd) cycles. Furthermore, it also hits 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 $\{u, v, z_{u,v}\}$ hit by a $z_{u,v} \in T$, can instead be hit by either $u$ or $v$ without increasing the size of $T$. Thus, we set $VC = T$.

By definition of an odd cycle transversal, $T$ hits 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 hit by $VC$. As $VC \cap Z = \emptyset$, $VC$ is a vertex cover of $G$.

The reduction $(f, g)$ is a Strict-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 $\gamma$. 

© Springer
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, \( G_{VC} \subset G_\gamma \) with \( \gamma = \frac{15}{2} \).

The following result is now immediate.

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

### 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 Sect. 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 \( m = 3 \) Dokka et al. (2014). 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 Fig. 3 where an example is depicted.

**Definition 4** – For any \( l \in \{0, \ldots, p+1\} \), \( i \in \{0, \ldots, m\} \), we define \( B^{(l, i)} = \{ v^i_j : v^i_j[l] = 0 \} \) 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 \( B^{(p+2, i)} = \{ v^i_j : v^i_j \text{ is a 1 vector} \} \). 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 \( \leq 2 \)) is completely characterized by the \( B^{(l, i)} \).

– For any \( l \in \{p+1\} \), the block \( B^l = \bigcup_{i \in \{m\}} B^{(l, i)} \).

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.

Notice that the vectors of a given block cannot “match” vectors of any other block, i.e., they don’t have their zero at the same position, thus any stack containing two vectors coming from different blocks will have at least two zeros in its representative vector. Based on this observation, the only relevant information, when assigning vectors of a given block, is not the list of vectors composing the “partial stacks” but the list of sets that are already “covered” by the “partial stacks”.

Indeed, to ensure that we are constructing a feasible solution, we only need to ensure that each stacks contains exactly one vector from each set and that no vector is contained in two different stacks. At each iteration, keeping tracks, for each of the \( n \) stacks under construction, of the sets containing a vector assigned to the stack allows us to construct a feasible solution. It remains to branch, for each block, on every possible assignment of the vectors of the blocks to get an optimal solution.
Definition 5

A partial stack $s$ of $I$ is a set of vectors $\{v_{x,i_1}, \ldots, v_{x,i_k}\}$ such that for all $x \in [k]$, $i_x \in [m]$, $i_x \neq i_{x'}$ for all $x, x' \in [k]$ satisfying $x \neq x'$ and such that for any $x \in [k]$, $v_{x,i_x} \in V^{i_x}$. 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 $c(s) = c(\bigwedge_{x \in [k]} v_{x,i_x})$ is the number of zeros of the bitwise AND of the vectors of $s$.

Informally, a partial stack is a stack that contains at most one vector per set. Its support is given by the indices of the sets containing a vector of the partial stack.

We define the notion of profile as follows:

Definition 6 A profile $P = \{x_c, \forall c \subseteq [m]\}$ is a set of $2^m$ non negative 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 7 Given two profiles $P = \{x_c : c \subseteq [m]\}$ and $P' = \{x'_{c'} : c' \subseteq [m]\}$ and a set $S = \{s_1, \ldots, s_n\}$ of $n$ partial stacks, $P'$ is said reachable from $P$ through $S$ iff there exist $n$ couples $(s_1, c_1), (s_2, c_2), \ldots, (s_n, c_n)$ such that:

 Springer
For each couple $(s, c)$, $\text{sup}(s) \cap c = \emptyset$.

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

For each $c' \subseteq [m]$, $|\{(s_j, c_j) : \text{sup}(s_j) \cup c_j = c', j = 1, \ldots, n\}| = x'_{c'}$. 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 Fig. 4.

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

**Optimization Problem 2 $\Pi$**

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

**Output** A set of $n$ partial stacks $S = \{s_1, s_2, \ldots, s_n\}$ such that $S$ is a partition of $B = \bigcup_{l \geq 1} B'$ 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 $(\min \sum_{0 \leq l \leq 1} 0)$ can be solved optimally by solving optimally the instance $I' = (1, P = \{x \emptyset = n, x_c = 0, \forall c \neq \emptyset\})$ of $\Pi$. 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 $\Pi$ by parsing the instance block after block and branching for each of these blocks on every reachable profile.
Function \texttt{MinSumZeroDP}(l, P)

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

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 When processing a given block \(l\), the algorithm tries every reachable profile. Since the zeros of vectors in blocks \(B = \bigcup_{l' < l} B^{l'}\) cannot be matched with those of vectors in block \(B' = \bigcup_{l' \geq l} B^{l'}\), branching on every reachable profile for each block allows us to enumerate every sets of representative vectors that can be constructed with the vectors of the instance. Note that two assignments leading to the same set of representative vectors share the same objective value. This proves the Lemma.

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 \(O^*(2^2m)\), the previous algorithm runs in \(O^*(n^{22m})\).

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 \(P\) 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 \texttt{Aux} in \texttt{MinSumZeroDP}, we get the following theorem.

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

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

In this article we mainly provided inapproximability results for \((\min \sum 0)^{\#0 \leq 1}\) that can be extended to the more general version of the problem \(\min \sum 0\). Though the main inapproximability result relies on UGC, such a result gives us strong clues on the hardness of approximation of \(\min \sum 0\) even if the ratio is allowed to depend on \(n\). On the approximability point of view, this provides complementary negative results to the state of the art that mainly focused on the parameters \(m\) and \(p\). The main open questions arising after this paper are the strengthening of the inapproximability results for \(\min \sum 0\) (with no restrictions on the number of zeros per vectors), by considering only \(\mathbf{P} \neq \mathbf{NP}\) hypothesis and the existence of an \(FPT\) algorithm for \((\min \sum 0)^{\#0 \leq 1}\) when parameterized by \(m\).

References

Alimonti P, Kann V (2000) Some APX-completeness results for cubic graphs. Theor Comput Sci 237(1–2):123–134
Bansal N, Khot S (2010) Inapproximability of hypergraph vertex cover and applications to scheduling problems. In: International Colloquium on Automata, Languages and Programming (ICALP), pp 250–261
Crescenzi P (1997) A short guide to approximation preserving reductions. In: Proceedings of the twelfth annual IEEE conference on computational complexity, Ulm, Germany, June 24–27, 1997, pp 262–273
Dokka T, Bougeret M, Boudet V, Giroudeau R, Spieksma FC (2013) Approximation algorithms for the wafer to wafer integration problem. In: Approximation and online algorithms (WAOA). Springer, pp 286–297
Dokka T, Crama Y, Spieksma FC (2014) Multi-dimensional vector assignment problems. Discrete Optim 14:111–125
Duvillié G, Bougeret M, Boudet V, Dokka T, Giroudeau R (2015) On the complexity of wafer-to-wafer integration. In: International conference on algorithms and complexity (CIAC), pp 208–220
Papadimitriou C, Yannakakis M (1988) Optimization, approximation, and complexity classes. In: Proceedings of the twentieth annual ACM symposium on theory of computing. ACM, pp 229–234
Reda S, Smith G, Smith L (2009) Maximizing the functional yield of wafer-to-wafer 3-d integration. IEEE Trans Very Large Scale Integr VLSI Syst 17(9):1357–1362
Vazirani VV (2001) Approximation algorithms. Springer, Berlin