On the complexity of Wafer-to-Wafer Integration

Guillerme Duvillié, Marin Bougeret, Vincent Boudet, Trivikram Dokka, Rodolphe Giroudeau

On the complexity of Wafer-to-Wafer Integration

G. Duvillié[1], M. Bougeret[1], V. Boudet[1],
T. Dokka[2], R. Giroudeau[1]

[1] LIRMM, Université Montpellier 2, France
{guillerme.duvillie,vincent.boudet,marin.bougeret,rodolphe.giroudeau}@lirmm.fr,
[2] Dept. of Management Science, Lancaster University
t.dokka@lancaster.ac.uk

Abstract. In this paper we consider the Wafer-to-Wafer Integration problem. A wafer is a p-dimensional binary vector. The input of this problem is described by m disjoints sets (called "lots"), where each set contains n wafers. The output of the problem is a set of n disjoint stacks, where a stack is a set of m wafers (one wafer from each lot). To each stack we associate a p-dimensional binary vector corresponding to the bit-wise AND operation of the wafers of the stack. The objective is to maximize the total number of "1" in the n stacks. We provide $O(m^{1-\epsilon})$ and $O(p^{1-\epsilon})$ non-approximability results even for $n = 2$, as well as a $\frac{p}{r}$-approximation algorithm for any constant r. Finally, we show that the problem is **FPT** when parameterized by p, and we use this **FPT** algorithm to improve the running time of the $\frac{p}{r}$-approximation algorithm.

1 Introduction

1.1 Problem definition

In this paper we consider Wafer-to-Wafer Integration problems. In these problems, we are given m disjoint sets $V^1 \dots V^m$, where each set V^i contains n binary p-dimensional vectors. For any $j \in [n]_1$ [1], and any $i \in [m]_1$, we denote by v^i_j the j^{th} vector of set V^i, and for any $k \in [p]_1$ we denote by $v^i_j(k) \in \{0,1\}$ the k^{th} component of v^i_j.

Let us now define the output. A stack $s = (v^s_1, \dots, v^s_m)$ is an $m-tuple$ of vectors such that $v^s_i \in V^i$, for any $i \in [m]_1$. An output of the problem is a set $S = \{s_1, \dots, s_n\}$ of n stacks such that for any i and j, vector v^i_j is contained exactly in one stack. An example of input and output is depicted in Figure 1.

These problems are motivated by an application in IC manufacturing in semiconductor industry, see [9] 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'. The objective of Wafer-to-Wafer Integration is to form n stacks, while maximizing the overall quality of the stacks (depending on the objective function).

[1] The notation $[n]_j$ stands for $\{j, \dots, n\}$.

Let us now define several objective functions, and the corresponding optimization problems. We consider the operator $\wedge$ which maps two p-dimensional vectors to another one by performing the logical *and* operation on each component of entry vectors. More formally, given two $p-$dimensional vectors u and v, we define $u \wedge v = (u(1) \wedge v(1), u(2) \wedge v(2), \ldots, u(p) \wedge v(p))$. We associate to any stack $s = (v_1^s, \ldots, v_m^s)$ a binary p-dimensional vector $v_s = \bigwedge_{i=1}^{m} v_i^s$. Then, the profit of a stack s is given by $c(v_s)$, where $c(v) = \sum_{k=1}^{p} v(k)$. Roughly speaking, the profit of a stack is the number of good bits in the stack, where a good bit (in position k) survives iff all the vectors of the stack have a good bit in position k.

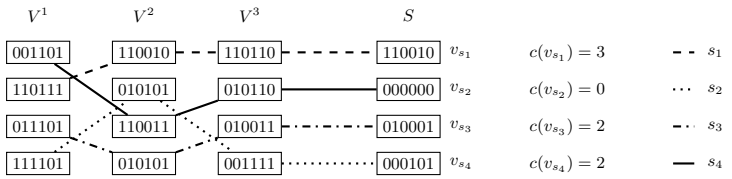

Fig. 1. Example of $\max \sum 1$ instance with $m = 3, n = 4, p = 6$ and of a feasible solution S of profit $f_{\Sigma 1}(S) = 7$.

We are now ready to define two following optimization problems:

Set of problems 1 $\max \sum 1$ and $\min \sum 0$	
Input	m sets of n binary p-dimensional vectors
Output	a set S of n disjoint stacks
Objective functions	$\max \sum 1$: maximize $f_{\sum 1}(S) = \sum_{j=1}^{n} c(v_{s_j})$, the total number of good bits $\min \sum 0$: minimize $f_{\sum 0}(S) = np - \sum_{j=1}^{n} c(v_{s_j})$, the number of bad bits

Instance of these problems will be denoted by $I[m, n, p]$. The notation $f(S)$ (instead of $f_{\sum 0}(S)$, $f_{\sum 1}(S), \ldots$) will be used when the context is non ambiguous.

1.2 Related work

In this paper we consider results in the framework of approximation and fixed parameter tractability theory. We only briefly recall the definitions here and refer the reader to [8,10] for more information. For any $\rho > 1$, a ρ-approximation algorithm A (for a maximization problem) is such that for any instance I, $A(I) \geq \frac{Opt(I)}{\rho}$, where $Opt(I)$ denotes the optimal value. The input of a parameterized (decision) problem Π is a couple (X, κ), where $X \subseteq \Sigma^*$ is a classical decision problem, and $\kappa : \Sigma^* \longrightarrow \mathbb{N}$ is a parameterization. Deciding Π requires to determine for any instance $I \in \Sigma^*$ if $I \in X$. Finally, we say that an algorithm A decides Π in **FPT** time (or that Π is **FPT** parameterized by κ) iff there exists a computable function f and a constant c such that for any I, $A(I)$ runs in $\mathcal{O}(f(\kappa(I))|I|^c)$.

The $\max \sum 1$ problem was originally defined in [9] as the "yield maximization problem in wafer-to-wafer 3-D integration technology". Authors of [9] point out that "the classical **NP**-hard 3-D matching problem is reducible to the $\max \sum 1$ problem". However, they do not provide the reduction and they only conclude that $\max \sum 1$ is **NP**-hard without stating consequences on the approximability. They also notice that $\max \sum 1$ is polynomial for $m = 2$ (as it reduces to finding a maximum profit perfect matching in a bipartite graph, solved by Hungarian Method), and design the "iterative matching heuristic" (IMH) that computes a solution based on (2D) matchings.

In [3] and [4] we investigated the $\min \sum 0$ problem by providing a $\frac{4}{3}$- approximation algorithm for $m = 3$ and several $f(m)$-approximation algorithms for arbitrary m (and for a more general profit function c). Furthermore, we also noticed in [3] that the natural ILP formulation implied that $\min \sum 0$ and $\max \sum 1$ are polynomial for fixed p. Concerning negative results, the implicit straightforward reduction from k-Dimensional Matching in [3] and made explicit in Appendix A, shows that $\min \sum 0$ is **NP**-hard, and $\max \sum 1$ is $\mathcal{O}(\frac{m}{\ln m})$ non-approximable. The more complex reduction of [4] shows that $\min \sum 0$ is **APX**-hard even for $m = 3$, and thus is very unlikely to admit a **PTAS**.

	[9]	[3], [4]	This paper
$\max \sum 1$	**NP**-hard	$\mathcal{O}(\frac{m}{\ln m})$ inapproximability polynomial for fixed p	for any ε, $\mathcal{O}(p^{1-\varepsilon})$ and $\mathcal{O}(m^{1-\varepsilon})$ inapproximability (even for $n = 2$) $\frac{p}{r}$-approximation in $\mathcal{O}(f(r)poly(n))$ **FPT**$/p$
$\min \sum 0$		for $m = 3 : \frac{4}{3}$-approximation, **APX**-hard $f(m)$-approximation for general m polynomial for fixed p	**FPT**$/p$

Table 1. Overview of results on Wafer-to-Wafer Integration

1.3 Contributions

In this paper we mainly study the $\max \sum 1$ problem, with a particular focus on parameter p. We prove in Subsection 2.2 that even for $n = 2$, for any ϵ, there is no $\rho(m, p)$-approximation algorithm for $\max \sum 1$ such that $\rho(x, x) = \mathcal{O}(x^{1-\epsilon})$ unless **P** = **NP** (this implies in particular no $\mathcal{O}(p^{1-\epsilon})$ and no $\mathcal{O}(m^{1-\epsilon})$ ratios). These negative results show that the simple p-approximation presented in Section 3.1 is somehow the best ratio we can hope for. Nevertheless, looking for better positive results we focus on $\frac{p}{r}$-approximation algorithm for any constant r. It turns out that any $\mathcal{O}(f(n, m, p))$ exact algorithm for $\max \sum 1$ can be used to derive a $\frac{p}{r}$-approximation in $\mathcal{O}(p \times f(n, m, r))$. This motivates our main result: determining the complexity of the $\max \sum 1$ problem when parameterized by p. The natural ILP in [4] implied that $\max \sum 1$ (and $\min \sum 0$) is polynomial for fixed p. In Section 3.2, we improve this result by showing that $\max \sum 1$ (and $\min \sum 0$) are **FPT** parameterized by p.

2 Negative Results

In order to obtain negative results for $\max \sum 1$, let us first introduce two related problems defined in the table Set of Problems 2.

Set of problems 2 $\max \max 1$ and $\max_{\neq 0}$	
Input	m sets of n binary p-dimensional vectors
Output	a set S of n disjoint stacks
Objective functions	$\max \max 1$: maximize $f_{\max 1}(S) = \max_{j \in [n]_1} c(v_{s_j})$, the profit of the best stack $\max_{\neq 0}$: maximize $f_{\neq 0}(S) = \lvert\{j \lvert c(v_{s_j}) \geq 1\}\rvert$, the number of non null stacks

Roughly speaking, we will see that approximating $\max \sum 1$ is *harder* than approximating these two problems, and that these problems are themselves non-approximable.

To show that approximability is preserved we will provide strict reductions [1] noted S-reductions. Indeed, if there is a strict reduction from Π_1 to Π_2, then any polynomial ρ-approximation for Π_2 yields to a ρ-approximation for Π_1. Notice that in the following we will rather provide reductions as defined in the following property.

Property 1 *Let Π_1 and Π_2 be two maximization problems with their given objective functions m_1 and m_2. Let f be a polynomial function that given any instance x of Π_1 associate an instance $f(x)$ of Π_2. Let g be a polynomial function that given any instance x of Π_1, and feasible solution S_2 of $f(x)$, associates a feasible solution $g(x, S_2)$ of Π_1. If f and g verify the two following conditions:*

1. $Opt(x) = Opt(f(x))$
2. $m_1(g(x, S_2)) \geq m_2(f(x))$

then (f, g) is a strict reduction.

2.1 Relation between $\max_{\neq 0}$, $\max \max 1$ and $\max \sum 1$

Observation 1 *There exists a strict reduction from* $\max \max 1$ *to* $\max \sum 1$.

Proof. Let us construct (f, g) as in Property 1. Consider an instance $I'[m', n', p']$ of $\max \max 1$. We construct an instance $I[m, n, p]$ of $\max \sum 1$ as follows: we set $p = p'$, $n = n'$, $m = m' + 1$. The m' sets of $I'[m', n', p']$ remain unchanged in $I[m, n, p]$: $\forall i \in [m']_1$, $V^i = V'^i$ and the last set $V^{m'+1}$ contains $(n-1)$ "zero vectors" (*i.e.* vectors having only 0) and one "one vector" (*i.e.* vector having only 1).

Informally, the set $V^{m'+1}$ of I behaves like a selecting mask: since all stacks except one are turned into zero stacks when assigning the vectors of last set, the unique "one vector" of set $V^{m'+1}$ must be assigned to the best stack, and maximizing the sum of the stacks is equivalent to maximizing the best stack.

More precisely, it is straightforward to see that the following statement is true: $\forall x$, $\exists$ *solution* S' *of* $\max\max 1$ *of value* $f_{\max 1}(S') = x \Leftrightarrow \exists$ *solution* S *of* $\max\sum 1$ *of value* $f_{\sum 1}(S) = x$. Thus, we get $Opt_{\max\max 1}(I') = Opt_{\max\sum 1}(I)$. As the previous reduction is polynomial, and a solution of I' can be deduced from a solution of I in polynomial time, we get the desired result. □

Observation 2 *There exists a strict reduction from* $\max_{\neq 0}$ *to* $\max\sum 1$.

We refer the reader to Appendix A for the reduction proving this lemma.

According to Observations 1 and 2, any non-approximability result for $\max_{\neq 0}$ or $\max\max 1$ will transfer to $\max\sum 1$. This motivates the next section.

2.2 Hardness of $\max_{\neq 0}$ and $\max\max 1$

The reduction from k-Dimensional Matching (kDM) provided in [3] can be adapted to $\max\sum 1$ instead of $\min\sum 0$ as shown in Appendix A Unlike the case of $\min\sum 0$, the reduction preserves approximability:

Theorem 1 (implicit in [3]). *There is a strict reduction from kDM to* $\max_{\neq 0}$.

As it is **NP**-hard to approximate kDM to a factor $\mathcal{O}(\frac{k}{ln(k)})$ [5], we get the following corollary:

Corollary 1. *It is* **NP***-hard to approximate* $\max_{\neq 0}$ *within a factor* $\mathcal{O}(\frac{m}{ln(m)})$.

We can also notice that any $\frac{m}{r}$-approximation ratio (for a constant $r \geq 3$) for $\max_{\neq 0}$ or $\max\sum 1$ would improve the currently best known ratio for kDM set to $\frac{k+1+\epsilon}{3}$ in [2].

Let us now consider a new reduction which provides results for $n = 2$ and according to parameter p.

Theorem 2. *There is a strict reduction from* Maximum Clique Problem *to* $\max\max 1$ *for* $n = 2$.

Proof. Let us construct (f, g) as in Property 1. Let us consider an instance $G = (V, E)$ of the Maximum Clique Problem. The corresponding instance of $\max\max 1$ is constructed as follows. We consider $m = |V|$ sets, each having two vectors. All the vectors have $p = |V|$ bits. For each vertex i of V, we create the set $V^i = (v_1^i, v_2^i)$. For any i, we define $v_1^i = (v_1^i(1), v_1^i(2), \ldots, v_1^i(p))$, where $v_1^i(k) = 1$ iff $\{i, k\} \in E$ or $i = k$, and $v_2^i = (v_2^i(1), v_2^i(2), \ldots, v_2^i(p))$, where $v_2^i(k) = 1$ iff $i \neq k$. In other words, v_1^i corresponds to the i^{th} row of the adjacency matrix of G, with a self loop.

The idea is that selecting v_1^i corresponds to selecting vertex i in graph, and selecting v_2^i will turn the i^{th} component to 0, which corresponds to a penalty for not choosing vertex i.

We first need to state an intermediate lemma. For any stack $s = \{v_1^s, \ldots, v_m^s\}$, let $X_s = \{i | v_i^s = v_1^i\}$ be the associated set of vertices in G. Recall that v_s is the p dimensional vector representing s.

Lemma 1. $\forall i \in [p]_1,\ v_s(i) = 1 \Leftrightarrow ((i \in X_s)$ *and* $(\forall x \in X_s \setminus i,\ \{x, i\} \in E))$.

$\triangle$ Let us first prove Lemma 1. Suppose i^{th} component of v_s is 1. This implies that $v_1^i \in s$, and thus $i \in X_s$. Now, suppose by contradiction that $\exists x \in X_s \setminus i$ such that $\{x,i\} \notin E$. $x \in X_s$ implies that $v_1^x \in s$. Moreover, $v_s(i) = 1$ implies that $v_1^x(i) = 1$, and thus $\{x,i\} \in E$, which leads to a contradiction. Suppose now that $i \in X_s$, and $\forall x \in X_s \setminus i,\ \{x,i\} \in E$. Let us prove that $\forall i'$, $v_{i'}^s(i) = 1$. Notice first that for $i' = i$ we have $v_i^s(i) = v_1^i(i) = 1$. Moreover, $\forall i' \neq i$ such that $i' \notin X_s$ we have $v_{i'}^s(i) = v_2^{i'}(i) = 1$. Finally, $\forall i' \neq i$ such that $i' \in X_s$, we have $v_{i'}^s(i) = v_1^{i'}(i) = 1$ as $\{i',i\} \in E$. $\triangle$

It is now straightforward to prove that $\forall x$, "$\exists$ solution S for $\max\max 1$ of value $f_{\max 1}(S) = x \Leftrightarrow \exists$ a clique X in G of size x." Indeed, suppose first that we have a solution S such that $f_{\max 1}(S) = x$. Let $s = (v_1^s, \ldots, v_m^s)$ be the stack in S of value x, and let $G_s = \{k | v_s(k) = 1\}$ be the set of good bits of s. We immediately get that the vertices corresponding to G_s form a clique in G, as $\forall i$ and $j \in G_s$ the previous property implies that $i \in X_s$, $j \in X_s$, and thus $\{i,j\} \in E$. Suppose now that there is a clique X^* in G, and let s be a stack such that $X_s = X^*$. The previous property implies that $\forall i \in X_s$, $v_s(i) = 1$.

Thus, $Opt_{\max\max 1}(I)$ is equal to the size of the maximum clique in G. As the previous reduction is polynomial, and as a solution of S of I can be translated back in polynomial time into the corresponding clique in G (of same size), we get the desired result. $\square$

As for any ϵ there is no $\mathcal{O}(|V|^{1-\epsilon})$-approximation for Maximum Clique Problem (with set of vertices V) unless **P**=**NP** [11], we get the following corollary:

Corollary 2. *Even for $n = 2$, for any ϵ, there is no $\rho(m,p)$-approximation such that $\rho(x,x) = \mathcal{O}(x^{1-\epsilon})$ for $\max\max 1$ and thus for $\max\sum 1$. Notice that in particular, $\mathcal{O}(p^{1-\epsilon})$ and $\mathcal{O}(m^{1-\epsilon})$ are not possible, but for example $(pm)^{\frac{1}{2}}$ is not excluded.*

To summarize, the main negative results for $\max\sum 1$ are no $\mathcal{O}(p^{1-\epsilon})$-approximation and no $\mathcal{O}(m^{1-\epsilon})$ approximation for $n = 2$, and no $\mathcal{O}(\frac{m}{\ln m})$-approximation for arbitrary n (using the reduction from k-Dimensional Matching of [4]). Notice that it does not seem obvious to adapt the previous reductions to provide the same non-approximability results for $\min\sum 0$. Thus, the question of improving the $f(m)$ ratios provided in [4] is still open.

3 Positive Results

In this Section, we develop some polynomial-time approximation algorithm for $\max\sum 1$. Then, we show that $\max\sum 1$ and $\min\sum 0$ are **FPT** parameterized by p.

3.1 $\frac{p}{r}$-approximation

Given the previous negative results, it seems natural to look for ratio $\frac{p}{r}$, where r is a constant. Let us first see how to achieve a ratio p with Algorithm 1.

Property 1. Algorithm 1 is a p-approximation algorithm for $\max\sum 1$.

Algorithm 1: p-approximation for $\max \sum 1$

$x = 0$;
while $\exists k$ *such that it is possible to create a stack* s *such that* $v_s(k) = 1$ **do**
    Add s to the solution;
    $x = x + 1$;
if $x < n$ **then**
    Add $n - x$ arbitrary (null) stacks to the solution;

Proof. Let $S = S_{\neq 0} \bigcup S_0$ be the solution computed by the algorithm, where $S_{\neq 0}$ is the set of non zero stacks, and S_0 is the set of the remaining null stacks. Since $S_{\neq 0}$ and S_0 are disjoint, we have $S_0 = S \setminus S_{\neq 0}$. Let $n_1 = |S_{\neq 0}|$, and $\forall i$, let $V'^i = V^i \bigcap S_0$. Let $n_2 = |S_0| = |V'^i|$ (all the V'^i have the same size). Notice that $n = n_1 + n_2$.

As the algorithm cannot create any non null stack at the end of the loop, we know that for any position $k \in [p]_1$, there is a set $i(k)$ such for any vector $w \in V'^{i(k)}$, $w(k) = 0$. In other words, we can say that there is a column of n_2 zeros in set $V'^{i(k)}$. Notice there may be several columns of zeros in a given set. Thus, we deduce that there are at least p columns (of n_2 zeros) in the vectors of $V'^{i(k)}$. Moreover, as none of these zeros can be matched in a solution, we know that these $n_2 p$ zeros will appear in any solution.

Thus, given S^* an optimal solution, we have $f(S^*) \leq np - n_2 p = n_1 p$. As $f(S) \geq n_1$, we get the desired result. □

Given a fixed integer r (and targeting a ratio $\frac{p}{r}$), a natural way to extend Algorithm 1 is to first look for r t-tuples (*i.e.* find $(k_1, \ldots, k_r)$ such that it is possible to create s such that $v_s(k_1) = \cdots = v_s(k_r) = 1$)), then $(r-1)$ t-tuple, *etc.* However, even for $r = 2$ this algorithm is not sufficient to get a ratio $\frac{p}{2}$, as shown by the example depicted in Figure2.

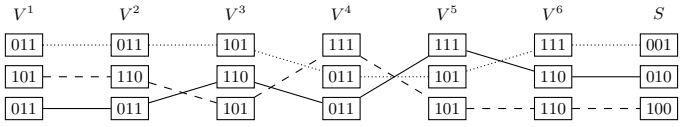

Fig. 2. Counter-example showing that Algorithm 1 for $r = 2$ remains a p-approximation. The depicted stacks correspond to an optimal solution of profit 3 whereas the algorithm outputs a solution of profit 1.

In this example it is not possible to create any stack of value strictly greater than 1 since set V^1 kills positions $\{1, 2\}$ (we say that a set kills positions $\{k_1, k_2\}$ iff there is no vector in the set such that $w(k_1) = w(k_2) = 1$), set V^2 kills positions $\{1, 3\}$, and set V^3 kills positions $\{2, 3\}$.

Thus, in this case (and more generally when no stack of value greater than 1 can be created), the solution computed by the algorithm for $r = 2$ is the same as one computed by Algorithm 1. In the worst case, the algorithm creates only one stack of value 1 (by choosing the first vector of each set). However, as depicted in Figure 2, the optimal value is 3, and thus the ratio $\frac{p}{2}$ is not verified.

In other words, knowing that no stack of profit 2 can be created does not provide better results for Algorithm 1. This motivates the different approach we follow hereafter.

Property 2. Suppose that there exists an exact algorithm for $\max \sum 1$ running in $f(n,m,p)$. Then, for any $r \in [p]_1$ we have a $\frac{p}{r}$-approximation running in $\mathcal{O}(p \times f(n,m,r))$.

Proof. The idea is to use a classical "shifting technique" by guessing the subset of the r most valuable consecutive positions in the optimal solution, and run the exact algorithm on these r positions.

Let S^* be an optimal solution for $\max \sum 1$. Let us write $f(S^*) = \sum_{k=1}^{p} a_k$, where $a_k = |\{s \in S^* | v_s(k) = 1\}|$ is the number of stacks in S^* that save position k. $\forall k \in [p-1]_0$, let $X_k = \{k, \ldots, (k+r-1) \ mod \ p\}$, and $\sigma_k = \sum_{t \in X_k} a_t$. Notice that we have $\sum_{k=1}^{p} \sigma_k = r \sum_{k=1}^{p} a_k = r f(S^*)$, as each value a_k appears exactly r times in $\sum_{k=1}^{p} \sigma_k$. This implies $\max_k \sigma_k \geq \frac{r}{p} f(S^*)$.

For any k, let I_k be the restricted instance where all the vectors are truncated to only keep positions in X_k (there are still nm vectors in I_k, but each vector is now a r dimensional vector). By running the exact algorithm on all the I_k and keeping the best solution, we get a $\frac{p}{r}$-approximation running in $\mathcal{O}(pf(n,m,r))$. □

The previous lemma motivates the exact resolution of $\max \sum 1$ in polynomial-time for fixed p. It is already proved in [4] that $\min \sum 0$ can be solved in $\mathcal{O}(m(n^{2^p}))$. As this result also apply to $\max \sum 1$, we get a $\frac{p}{r}$-approximation running in $\mathcal{O}(pm(n^{2^r}))$, for any $r \in [p]_1$. Our objective is now to improve this running time by showing that $\max \sum 1$ (and $\min \sum 0$) are even **FPT** parameterized by p (and not only polynomial for fixed p).

3.2 Faster algorithm for fixed p for $\max \sum 1$

Definition 1. *For any $t \in [2^p - 1]_0$, we define configuration t as B_t: the p-dimensional binary vector that represents t in binary. We say that a p-dimensional vector v is in configuration t iff $v = B_t$.*

First ideas to get an **FPT** *algorithm*

Let us first recall our previous algorithm in [4] for fixed p. This result is obtained using an integer linear programming formulation of the following form. The objective function is $\min \sum_{t=0}^{2^p-1} x_t \bar{c}_t$ (recall that in [4] the considered objective function is $\min \sum 0$), where $x_t \in [n]_0$ is an integer variable representing the number of stacks in configuration t, and $\bar{c}_t \in [p]_0$ is the number of 0 in configuration t.

This is a good starting point to get an **FPT** algorithm. Indeed, if we note n_{var} (resp. m_{ctr}) the number of variables (resp. number of constraints) of an ILP, for any $A \in \mathbb{Q}^{n_{var} \times m_{ctr}}, b \in \mathbb{Q}^{m_{ctr}}$, the famous algorithm of Lenstra [7] allows us to decide the feasibility of an ILP, under the form $\exists ? x \in \mathbb{Z}^{n_{var}} | Ax \leq b$, in time $\mathcal{O}(C^{n_{var}^3} m_{ctr}^{O(1)})$ (this running time is given in [6]), where C is a constant. Thus, to get an **FPT** algorithm parameterized by p, it is sufficient to write $\min \sum 0$ (and $\max \sum 1$) as an ILP using $f(p)$ variables.

However, it remains now to add constraints that represent the $\min \sum 0$ problem. In [4], these constraints are added using z^i_{jt} variables (for $i \in [m]_1, j \in [n]_1, t \in [2^p - 1]_0)$), where $z^i_{jt} = 1$ iff v^i_j is assigned to a stack of type t. Nevertheless these new $\mathcal{O}(mn2^p)$ variables prevent us to use [7]. Thus, we now come back to the $\max \sum 1$ problem, and our objective is to express the constraints using only the $\{x_t\}$ variables.

Presentation of the new ILP for $\max \sum 1$

For any $t \in [2^p - 1]_0$, we define an integer variable $x_t \in [n]_0$ representing the number of stacks in configuration t. Let also $c_t \in [p]_1 = c(B_t)$ be the number of 1 in configuration t.

Definition 2. *A profile is a tuple* $P = \{x_0, \ldots, x_{2^p-1}\}$ *such that* $\sum_{t=0}^{2^p-1} x_t = n$.

Definition 3. *The profile* $Pr(S) = \{x_0, \ldots, x_{2^p-1}\}$ *of a solution* $S = \{s_1, \ldots, s_n\}$ *is defined by* $x_t = |\{i | v_{s_i}$ *is in configuration* $t\}|$, *for* $t \in [2^p - 1]_0$.

Definition 4. *Given a profile* P, *an associated solution* S *is a solution such that* $Pr(S) = P$. *We say that a profile* P *is feasible iff there exists an associated solution* S *that is feasible.*

Notice that the definition of associated solutions also applies to a non feasible profile. In this case, any associated solution will also be non feasible.

Obviously, the $\max \sum 1$ problem can be formulated using the following ILP:

$$\max \quad \sum_{t=0}^{2^p-1} x_t c_t \quad \text{subject to} \quad \sum_{t=0}^{2^p-1} x_t = n$$
$$\forall 0 \leq t < 2^p, \ x_t \in \mathbb{N}$$
$$P = \{x_t\} \text{ is feasible}$$

Our objective is now to express the feasibility of a profile by using only these 2^p variables. Roughly speaking, the idea to ensure the feasibility is the following. Let us suppose (with $p = 2$ and $n = 4$ for example) that there exists a feasible solution of fixed profile $x_0 = 0, x_1 = 1, x_2 = 2, x_3 = 1$. Suppose also that the first set is as depicted in Figure 3. To create a feasible solution with this profile, we have to "satisfy" (for each set V^i) the demands x_t for all configurations t. For example in set 1, the demand x_2 can be satisfied by using one vector in configuration 2 and one vector of configuration 3, and the demand 3 can be satisfied using the remaining vector of 3 (the demand x_0 is clearly satisfied). Notice that a demand of a given configuration (*e.g.* configuration 2 here) can be satisfied using a vector that "dominates" this configuration (*e.g.* configuration 3 here). The notion of domination will be introduced in Definition 5. Thus, a feasible profile implies that for any set i there exists a perfect matching between the vectors of V^i and the profile $\{x_t\}$.

Let us now define more formally the previous ideas.

Definition 5 (Domination).

A p-dimensional vector v_1 *dominates a p-dimensional vector* v_2 *(denoted by* $v_1 \gg v_2$*) iff* $\forall k \in [p]_1, \ v_2(k) = 1 \Rightarrow v_1(k) = 1$.

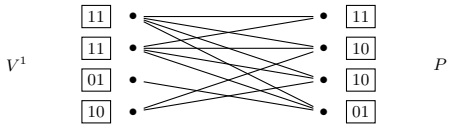

Fig. 3. Example showing that satisfying demands of profile P with set 1 requires to find a perfect matching. Edges represent domination between configuration.

A configuration $t_1 \in [2^p - 1]_0$ dominates a configuration $t_2 \in [2^p - 1]_0$ (denoted by $t_1 \gg t_2$) iff $B_{t_1} \gg B_{t_2}$ (recall that B_t is the p-dimensional binary representation of t).

A solution S' dominates a solution S (denoted by $S' \gg S$) iff $\exists$ a bijection $\phi : [n]_1 \to [n]_1$ such that for any $i \in [n]_1$, $v_{s'_i} \gg v_{s_{\phi(i)}}$ (in other word, there is a one to one domination between stacks of S' and stacks of S).

A profile P' dominates a profile P (denoted by $P' \gg P$) iff there exists solutions S' and S such that $Pr(S') = P'$, $Pr(S) = P$ and $S' \gg S$.

Definition 6. *For any $i \in [m]_1$ and any $t \in [2^p - 1]_0$, let b_t^i be the number of vectors of set V^i in configuration t.*

Definition 7 (Graph G_P^i).

Let P be a profile not necessarily feasible. Let $G_P^i = ((\Delta_P, \Lambda^i), E_{\gg})$, where $\Lambda^i = \{\lambda_t^{i,l}, 0 \leq t \leq 2^p - 1, 1 \leq l \leq b_t^i\}$, and $\Delta_P = \{\delta_t^l, 0 \leq t \leq 2^p - 1, 1 \leq l \leq x_t\}$. Let us fix a bijection $f : \Delta_P \cup \Lambda^i \mapsto [2^p - 1]_0$, that associates to each vertex $\lambda_t^{i,l}$ and to each vertex δ_t^l the vector in configuration t. Λ^i (resp. Δ_P) represents the set of vectors of V^i (resp. the demands of profile P) grouped according to their configurations. Notice that $|\Lambda^i| = |\Delta_P| = n$. Finally, we set $E_{\gg} = \{\{a, b\} | a \in \Delta_P, b \in \Lambda^i, f(a) \ll f(b)\}$.

We are now ready to show the following proposition.

Proposition 1. *For any profile $P = \{x_0, \ldots, x_{2^p-1}\}$,*

$$(\exists P' \text{ feasible, with } P' \gg P) \Leftrightarrow \forall i \in [m]_1, \exists \text{a matching of size } n \text{ in } G_P^i$$

Before starting the proof, notice that the simpler proposition "for any P, P feasible $\Leftrightarrow \forall i \in [m]_1$, there is a matching of size n in G_P^i" does not hold. Indeed, $\Rightarrow$ is correct, but $\Leftarrow$ is not: consider P with $x_0 = n$ (recall that configuration 0 is the null vector), and an instance with nm "1 vectors" (containing only 1). In this case, there is a matching of size n in all the G_P^i, but P is not feasible. This explains the formulation of Proposition 1. An example of the correction formulation is depicted Figure 4.

Proof. Let P be a profile.

($\Rightarrow$) Let P' be a feasible profile that dominates P. Let $S = \{s_1, \ldots, s_n\}$ and $S' = \{s'_1, \ldots, s'_n\}$ two solutions such that S' is feasible, $Pr(S) = P$, $Pr(S') = P'$ (notice that S and P are not necessarily feasible), and $S' \gg S$. Without loss of generality, let us assume that $\forall j$, $s'_j \gg s_j$ (*i.e.* the bijection ϕ of Definition 5 is the identity), and let us assume that for any j, $s'_j = (v_j^1, \ldots, v_j^m)$. Since $v_j^i \in s'_j$,

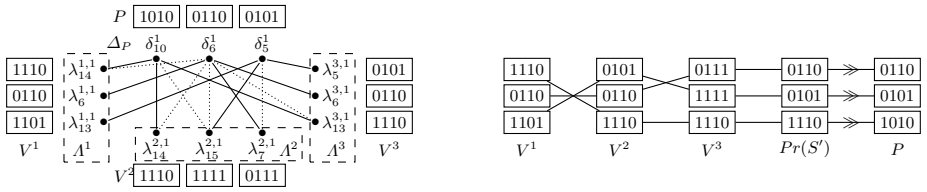

Fig. 4. Illustration of Proposition 1 with $m = n = 3$ and $p = 4$. Left: The three G_P^i graphs (edges are depicted by solid and dotted lines), and three matchings (in solid lines) corresponding to S'. Right: Solution S' s.t. $Pr(S') \gg P$.

then for any i, we know that $v_j^i \gg s'_j \gg s_j$, $\forall j \in [n]_1$. This implies a matching of size n in all the graphs G_P^i.

($\Leftarrow$) Let us suppose that $\forall i \in [m]_1$, there is a matching $\mathcal{M}^i$ of size n in G_P^i.

W.l.o.g.let us rename $\{\delta_1, \ldots, \delta_n\}$ the vertices of Δ_P, and $\{\lambda_1^i, \ldots, \lambda_n^i\}$ the vertices of Λ^i such that for any i, $\mathcal{M}^i = \{\{\lambda_1^i, \delta_1\}, \ldots, \{\lambda_n^i, \delta_n\}\}$. This implies $f(\lambda_1^i) \gg f(\delta_1)$, $\ldots$, $f(\lambda_n^i) \gg f(\delta_n)$. Let us define $S = \{s_1, \ldots, s_n\}$, where $\forall j \in [n]_1$, $s_j = (f(\lambda_j^1), \ldots, f(\lambda_j^m))$. Notice that for any j, $s_j \gg f(\delta_j)$, as all the $f(\lambda_j^i) \gg f(\delta_j)$, and combining two vectors $f(\lambda_j^{i_1}) \gg f(\delta_j)$ and $f(\lambda_j^{i_2}) \gg f(\delta_j)$ creates another vector that dominates $f(\delta_j)$. Thus, S is feasible, and $Pr(S) \gg P$, and we set $P' = Pr(S)$. □

Now, we can use the famous Hall's Theorem to express the existence of a matching in every set.

Theorem 3 (Hall's Theorem). *Let $G = ((V^1, V^2), E)$ a bipartite graph with $|V^1| = |V^2| = n$. There is a matching of size n in G iff $\forall \sigma \subseteq V^1$, $|\sigma| \leq |\Gamma(\sigma)|$, where $\Gamma(\sigma) = \{v_2 \in V^2 | \exists v_1 \in \sigma$ such that $\{v_1, v_2\} \in E\}$.*

Remark 1. Notice that we cannot use Hall's Theorem directly on graphs G_P^i, as we would have to add the 2^n constraints of the form $\forall S \subseteq V^i$. However, we will reduce the number of constraints to a function $f(p)$ by exploiting the particular structure of G_P^i.

Proposition 2 (Matching in G_P^i).

$\forall i \in [m]_1$, $\forall P = \{x_0, \ldots, x_{2^p-1}\}$*:*
$(\forall \sigma \subseteq \Delta_P,\ |\sigma| \leq |\Gamma(\sigma)|) \Leftrightarrow (\forall \sigma_{cfg} \subseteq [2^p - 1]_0, \sum_{t \in \sigma_{cfg}} x_t \leq \sum_{t \in dom(\sigma_{cfg})} b_t^i)$ *where $dom(\sigma_{cfg}) = \{t' | \exists t \in \sigma_{cfg}$ such that $t' \gg t\}$ is the set of configurations that dominate σ_{cfg}.*

Proof. ($\Rightarrow$) Let $\sigma_{cfg} = \{t_1, \ldots, t_\alpha\}$. Let $\sigma = \{\delta_{t_i}^l, 1 \leq i \leq \alpha, 1 \leq l \leq x_{t_i}\}$ be the vertices of Δ_P corresponding to the demands in σ_{cfg}. Observe that $\sum_{t \in \sigma_{cfg}} x_t = |\sigma|$. Notice also that $\Gamma(\sigma) = \{\lambda_t^{i,l}, t \in dom(\sigma), 1 \leq l \leq b_t^i\}$ by construction. Thus, $|\sigma| \leq |\Gamma(\sigma)|$ implies $\sum_{t \in \sigma_{cfg}} x_t \leq \sum_{t \in dom(\sigma_{cfg})} b_t^i$.

($\Leftarrow$) Let $\sigma \subseteq \Delta_P$. $\forall t \in [2^p - 1]_0$, let $X_t = \{\delta_t^l, 1 \leq l \leq x_t\}$, let $\sigma_t = \sigma \bigcap X_t$. Let $\sigma_{cfg} = \{t_1, \ldots, t_\alpha\} = \{t | \sigma_t \neq \emptyset\}$. Let $X = \bigcup_{t \in \sigma_{cfg}} \{X_t\}$. Notice that $|\sigma| \leq |X| = \sum_{t \in \sigma_{cfg}} x_t$.

Let us first prove that $\Gamma(\sigma) = \Gamma(X)$. $\Gamma(\sigma) \subseteq \Gamma(X)$ is obvious. Now, if there is a $\lambda_{t'}^{i,l'} \in \Gamma(X)$, it means that there is a $t \in \sigma_{cfg}$ such that $\lambda_{t'}^{i,l'} \in \Gamma(X_t)$, and

thus there exists l such that $\{\delta_t^l, \lambda_{t'}^{i,l'}\} \in E$ (which implies that $t' \gg t$). As $\sigma_t \neq \emptyset$, there exists l' such that $\delta_t^{l'} \in \sigma_t$, and $\{\delta_t^{l'}, \lambda_{t'}^{i,l'}\} \in E$ as $t' \gg t$.

Finally, the hypothesis with our set σ_{cfg} leads to

$$|\sigma| \leq |X| = \sum_{t \in \sigma_{cfg}} x_t \leq \sum_{t \in dom(\sigma_{cfg})} b_t^i = |\Gamma(X)| = |\Gamma(\sigma)|$$

Using Propositions 1 and 2, we can now write that for any profile $P = \{x_0, \ldots, x_{2^p-1}\}$: □

$$\exists P' \text{ feasible, with } P' \gg P \Leftrightarrow \forall i, \forall \sigma_{cfg} \subseteq [2^p - 1]_0, \sum_{t \in \sigma_{cfg}} x_t \leq \sum_{t \in dom(\sigma_{cfg})} b_t^i.$$

Thus, we use now the following ILP to describe the $\max \sum 1$ problem:

$$\begin{aligned} \max \quad & \sum_{t=0}^{2^p-1} x_t c_t \\ \text{subject to} \quad & \forall i \in [m]_1 : \forall \sigma_{cfg} \subseteq [2^p - 1]_0, \sum_{t \in \sigma_{cfg}} x_t \leq \sum_{t \in dom(\sigma_{cfg})} b_t^i \\ & \forall t \in [2^p - 1]_0, x_t \in \mathbb{N} \end{aligned}$$

This linear program has 2^p variables and $(m2^{2^p} + 2^p)$ constraints. Thus, we can solve it using [7] in time $f(p)poly(n+m)$, we get that $\max \sum 1$ and $\min \sum 0$ are **FPT** parameterized by p. Using Property 2 this ILP leads to a $\frac{p}{r}$-approximation algorithm for $\max \sum 1$ running in time $f(r)poly(n+m)$.

4 Conclusion

In this article, we established that $\max \sum 1$ is $\mathcal{O}(m^{1-\varepsilon})$ and $\mathcal{O}(p^{1-\varepsilon})$ non-approximable for $n = 2$. On the positive side, we provided a **FPT** algorithm for $\max \sum 1$ leading to a $\frac{p}{r}$-approximation algorithm running in $\mathcal{O}(f(r)poly(m+n))$, which is the best we can hope for. The existence of a $\rho(m)$-approximation (typically m) algorithm remains open.

References

1. P. Crescenzi, V. Kann, R. Silvestri, and L. Trevisan. Structure in approximation classes. *SIAM Journal on Computing*, 28(5):1759–1782, 1999.
2. M. Cygan. Improved approximation for 3-dimensional matching via bounded pathwidth local search. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 509–518. IEEE, 2013.
3. T. Dokka, M. Bougeret, V. Boudet, R. Giroudeau, and F. CR Spieksma. Approximation algorithms for the wafer to wafer integration problem. In *Approximation and Online Algorithms (WAOA)*, pages 286–297. Springer, 2013.
4. T. Dokka, Y. Crama, and F. C.R. Spieksma. Multi-dimensional vector assignment problems. *Discrete Optimization*, 14:111–125, 2014.
5. E. Hazan, S. Safra, and O. Schwartz. On the complexity of approximating k-dimensional matching. In *Approximation, Randomization, and Combinatorial Optimization.. Algorithms and Techniques*, pages 83–97. Springer, 2003.

6. S. Kratsch. On polynomial kernels for integer linear programs: Covering, packing and feasibility. In *Algorithms–ESA 2013*, pages 647–658. Springer, 2013.
7. H. W. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of operations research*, 8(4):538–548, 1983.
8. R. Niedermeier. Invitation to fixed-parameter algorithms. 2006.
9. 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.
10. D. P. Williamson and D. B. Shmoys. *The design of approximation algorithms*. Cambridge University Press, 2011.
11. D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 681–690. ACM, 2006.

A Reduction from kDM

In this section, we provide the reductions proving that $\max \sum 1$ is *harder* than $\max_{\neq 0}$ and that $\max_{\neq 0}$ is even *harder* than kDM.

Observation 2 *There exists a strict reduction from* $\max_{\neq 0}$ *to* $\max \sum 1$.

Proof. Consider an instance $I'[m', n', p']$ of $\max_{\neq 0}$. We construct an instance $I[m, n, p]$ of $\max \sum 1$ as follows. The number of components of each vector is left unchanged ($p = p'$), the number of vectors per set is multiplied by p' ($n = n'p'$) and the number of sets is increased by one ($m = m' + 1$). $\forall j = 1, \ldots, m'$, the sets V^j are constructed as follows: $V^i = V'^i \bigcup X$, where X contains $n - n'$ null vectors, and $V^{m'+1}$ contains n' times the following sets of vectors (this is the reason why $n = n'p'$):

$$\{\underbrace{1000\ldots000}_{p=p'}, 0100\ldots000, 0010\ldots000, \ldots, 0000\ldots010, 0000\ldots001\}$$

As an example, the following instance $I'[3, 2, 4]$ of $\max_{\neq 0}$

$$\begin{array}{lll} V'^1_1 = 1010 & V'^2_1 = 0001 & V'^3_1 = 1111 \\ \underbrace{V'^1_2 = 1001}_{V^1} & \underbrace{V'^2_2 = 0100}_{V^2} & \underbrace{V'^3_2 = 1000}_{V^3} \end{array}$$

is turned into the following one $I[4, 8, 4]$ of $\max \sum 1$:

$$\begin{array}{llll} v^1_1 = 1010 & v^2_1 = 0001 & v^3_1 = 1111 & v^4_1 = 1000 \\ v^1_2 = 1001 & v^2_2 = 0100 & v^3_2 = 1000 & v^4_2 = 0100 \\ v^1_3 = 0000 & v^2_3 = 0000 & v^3_3 = 0000 & v^4_3 = 0010 \\ v^1_4 = 0000 & v^2_4 = 0000 & v^3_4 = 0000 & v^4_4 = 0001 \\ v^1_5 = 0000 & v^2_5 = 0000 & v^3_5 = 0000 & v^4_5 = 1000 \\ v^1_6 = 0000 & v^2_6 = 0000 & v^3_6 = 0000 & v^4_6 = 0100 \\ v^1_7 = 0000 & v^2_7 = 0000 & v^3_7 = 0000 & v^4_7 = 0010 \\ \underbrace{v^1_8 = 0000}_{V^1} & \underbrace{v^2_8 = 0000}_{V^2} & \underbrace{v^3_8 = 0000}_{V^3} & \underbrace{v^4_8 = 0001}_{V^4} \end{array}$$

Informally, as the set V^m of I turns any non zero stack of I' into a stack of value 1 (by choosing an appropriate vector), maximizing the total number of 1 in I requires to maximize the number of non null stacks in I'.

Let us check first that "$\forall$ solution S' of $\max_{\neq 0}$, $\exists$ solution S of $\max \sum 1$ of value $f_{\sum 1}(S) = f_{\neq 0}(S')$". Let $\{s'_1, \ldots, s'_x\}$ be the x non null stacks of S', and $\{s'_{x+1}, \ldots, s'_{n'}\}$ be the null stacks of S'. Let us now construct S. For any $i, 1 \leq i \leq x$, let k_i be a non null bit in s'_i. We extend s'_i to a stack s_i by adding a vector v^m_j of set m such that $v^m_j(k_i) = 1$. Notice that such a vector always exists as for any position $k, 1 \leq k \leq p$ there are n' wafers in set m whose bit in position k is equal to 1. Thus, even if the x stacks of S' have the same non null position k, the previous construction is possible. Finally, the $n' - x$ remaining null stacks of S' are extended arbitrarily, and we complete the construction of S by adding $n - n'$

arbitrary stacks (as these stacks use in each of the first $m-1$ set the set X of null vectors, the value of these stacks is zero). Thus, we get $f_{\sum 1}(S) = x$.

Let us new check that "$\forall$ solution S of $\max \sum 1$, $\exists$ solution S' of $\max_{\neq 0}$ of value $f_{\max 1}(S') \geq f_{\neq 0}(S)$. As any vector of set m has only one good bit (*i.e.*equal to 1), the profit of any stack of S is at most 1, and thus there is exactly x non null stacks $\{s_1, \ldots, s_x\}$ in S. By removing the vector in set m in each of these x stacks, we get x non null stacks of I'. Finally, we complete the construction of S' by creating arbitrarily the $n' - x$ remaining stacks, and we get $f_{\neq 0}(S') \geq x$ (notice that the value of S' can be greater than x, as we could have a null stack $s_i \in S$ whose restriction to the first $(m-1)$ set is a non null stack of I').

Thus, we get $Opt_{\max_{\neq 0}}(I') = Opt_{\max \sum 1}(I)$. As the previous reduction is polynomial, and as a solution of S of I can be translated back in polynomial time into a solution S' of I' with $f_{\neq 0}(S) \geq f_{\sum 1}(S')$, we get the desired result.

□

We now prove that $\max_{\neq 0}$ is *harder* than kDM. But first of all, let us recall the k Dimensional Matching problem.

Input	k pairwise disjoint sets of n elements and x $k-$tuple $t_l \in X_1 \times \cdots \times X_k, 1 \leq l \leq x$
Output	a set of disjoint $k-$tuples
Objective functions	maximize the number of tuples in solution

Theorem 2 (implicit in [3]). *For any $\rho(m)$, any polynomial $\rho(m)$-approximation algorithm for $\max_{\neq 0}$ can be converted into a polynomial $\rho(k)$-approximation algorithm for the kDM problem.*

Proof. Let us consider an instance of kDM described by k sets $X_i, 1 \leq i \leq k$ (where X_i are pairwise disjoint) such that $|X_i| = n$, and x k-tuple $t_l \in X_1 \times \cdots \times X_k, 1 \leq l \leq x$. We denote by $a_i^j, 1 \leq j \leq n$ the elements of set X_i. From this instance, we construct an instance of $\max_{\neq 0}$ composed of k sets, each containing n vectors. The number of bits per vector is equal to x. Vector j of set i represents the set of k-tuple that use element a_i^j. Thus, we define v_j^i as a string of size x, where the k^{th} bit is set to 1 iff a_i^j is used in t_k. Thus, the k^{th} bit of a stack is 1 iff each element of tuple k is selected (by selecting corresponding vector), and then iff tuple t belongs to solution of kDM instance. Notice that the value of any stack is at most 1, since a stack represents a tuple. □