# Deterministic Algorithms for Matching and Packing Problems Based on Representative Sets*

PRACHI GOYAL†, NEELDHARA MISRA‡, FAHAD PANOLAN§, AND MEIRAV ZEHAVI¶

**Abstract.** In this work, we study the well-known $r$-Dimensional $k$-Matching ($(r,k)$-DM) and $r$-Set $k$-Packing ($(r,k)$-SP) problems. Given a universe $U := U_1 \uplus \cdots \uplus U_r$, and an $r$-uniform family $\mathcal{F} \subseteq U_1 \times \cdots \times U_r$, the $(r,k)$-DM problem asks if $\mathcal{F}$ admits a collection of $k$ mutually disjoint sets. Given a universe $U$, and an $r$-uniform family $\mathcal{F} \subseteq 2^U$, the $(r,k)$-SP problem asks if $\mathcal{F}$ admits a collection of $k$ mutually disjoint sets.

We employ techniques based on dynamic programming and representative families. This leads to a deterministic algorithm with running time $\mathcal{O}(2.851^{(r-1)k} \cdot |\mathcal{F}| \cdot n \log^2 n \cdot \log W)$ for the weighted version of $(r,k)$-DM, where $W$ is the maximum weight in the input, and a deterministic algorithm with running time $\mathcal{O}(2.851^{(r-0.5501)k} \cdot |\mathcal{F}| \cdot n \log^2 n \cdot \log W)$ for the weighted version of $(r,k)$-SP. Thus, we significantly improve the previous best known deterministic running times for $(r,k)$-DM and $(r,k)$-SP, and the previous best known running times for their weighted versions. We rely on structural properties of $(r,k)$-DM and $(r,k)$-SP to develop algorithms that are faster than those that can be obtained by a standard use of representative sets. Incorporating the principles of iterative expansion, we obtain a better algorithm for $(3,k)$-DM, running in time $\mathcal{O}(2.004^{3k} \cdot |\mathcal{F}| \cdot n \log^2 n)$. We believe that this algorithm demonstrates an interesting application of representative families in conjunction with more traditional techniques. Furthermore, we present kernels of size $\mathcal{O}(e^r r(k-1)^r \log W)$ for the weighted versions of $(r,k)$-DM and $(r,k)$-SP, improving the previous best known kernels of size $\mathcal{O}(r!r(k-1)^r \log W)$ for these problems.

**Key words.** $r$-Dimensional Matching, Set Packing, 3D-Matching, Fixed-Parameter Algorithms, Representative Sets, Iterative Expansion

## 1. Introduction.

Given a universe $U := U_1 \uplus \cdots \uplus U_r$, and an $r$-uniform family $\mathcal{F} \subseteq U_1 \times \cdots \times U_r$, the $r$-Dimensional $k$-Matching ($(r,k)$-DM) problem asks if $\mathcal{F}$ admits a collection of $k$ mutually disjoint sets. The special case in which $r = 3$ can be viewed as an immediate generalization of the matching problem on bipartite graphs to three-partite, three-uniform hypergraphs. The question of finding the largest 3D-Matching is a classic optimization problem, and the decision version is listed as one of the six fundamental NP-complete problems in Garey and Johnson [18]. The $(r,k)$-DM problem may be thought as a restricted version of the more general $r$-Set $k$-Packing ($(r,k)$-SP) problem, where no restrictions are assumed on the universe. More precisely, given a universe $U$, and an $r$-uniform family $\mathcal{F} \subseteq 2^U$, the $(r,k)$-SP problem asks if $\mathcal{F}$ admits a collection of $k$ mutually disjoint sets. In this paper, we study the parameterized complexity of $(r,k)$-DM and $(r,k)$-SP.

*A preliminary version of this paper appeared in the proceedings of FSTTCS 2013 [19]

†Indian Institute of Science, Bangalore, India (prachi.goyal@csa.iisc.ernet.in)

‡Indian Institute of Science, Bangalore, India (neeldhara@csa.iisc.ernet.in)

§Institute of Mathematical Sciences, Chennai, India (fahad@imsc.res.in)

¶Technion - Israel Institute of Technology, Haifa, Israel (meizeh@cs.technion.ac.il)

Neeldhara Misra is supported by the DST-INSPIRE fellowship, project DSTO-1209.

| Reference | Problem | Weighted? | Algorithm | Running Time |
|---|---|---|---|---|
| Chen *et al.* [6] | SP | No | D | $\mathcal{O}^*((rk)^{O(rk)})$ |
| Downey *et al.* [13] | SP | Yes | D | $\mathcal{O}^*((rk)^{O(rk)})$ |
| Fellows *et al.* [14] | SP | Yes | D | $\mathcal{O}^*(2^{O(rk)})$ |
| Koutis [21] | SP | No | D | $\mathcal{O}^*(2^{O(rk)})$ |
| | SP | No | R | $\mathcal{O}^*(10.874^{rk})$ |
| Chen *et al.* [9] | SP | No | D | $\mathcal{O}^*(5.44^{rk})$ |
| | DM | No | D | $\mathcal{O}^*(5.44^{(r-1)k})$ |
| Chen *et al.* [7] | SP | Yes | D | $\mathcal{O}^*(4^{rk+o(rk)})$ |
| | SP | Yes | R | $\mathcal{O}^*(4^{(r-1)k+o(rk)})$ |
| Chen *et al.* [5] | SP | Yes | D | $\mathcal{O}^*(4^{(r-0.5)k+o(rk)})$ |
| | DM | Yes | D | $\mathcal{O}^*(4^{(r-1)k+o(rk)})$ |
| Koutis [22] | SP | No | R | $\mathcal{O}^*(2^{rk})$ |
| Koutis *et al.* [23] | DM | No | R | $\mathcal{O}^*(2^{(r-1)k})$ |
| Björklund *et al.* [2] | DM | No | R | $\mathcal{O}^*(2^{(r-2)k})$ |
| This work | SP | Yes | D | $\mathcal{O}^*(2.851^{(r-0.5501)k})$ |
| | DM | Yes | D | $\mathcal{O}^*(2.851^{(r-1)k})$ |

TABLE 1

*Algorithms for $(r,k)$-DM and $(r,k)$-SP, with D denoting deterministic algorithms and R denoting randomized algorithms.*

**1.1. Prior Work. Algorithms for $(r,k)$-DM and $(r,k)$-SP.** The questions of $(r,k)$-DM and $(r,k)$-SP, including their weighted versions, have enjoyed substantial attention in the context of exact parameterized algorithms, and there have been several deterministic and randomized approaches to these problems (see Table 1).[I]

One of the earliest approaches [13] used the color-coding technique [1]. Further, in [14], a kernel was developed for $(r,k)$-SP, and the color-coding was combined with dynamic programming on the structure of the kernel to obtain an improvement. In [21], Koutis used an algebraic formulation of $(r,k)$-SP and proposed a randomized algorithm (derandomized with hash families). The randomized approaches saw further improvements in subsequent work [22, 23, 2], also based on algebraic techniques. The common theme in these developments is to express a parameterized problem in an algebraic framework by associating monomials with the combinatorial structures that are sought, ultimately arriving at a multilinear monomial testing problem or a polynomial identity testing problem. In a recent development [9], a derandomization method was proposed for these algebraic approaches, leading to deterministic algorithms that solve $(r,k)$-SP and $(r,k)$-DM in times $\mathcal{O}^*(5.44^{rk})$ and $\mathcal{O}^*(5.44^{(r-1)k})$, respectively.

Prior to our work, the algorithms having the best deterministic running times for $(r,k)$-DM and $(r,k)$-SP were those of Chen *et al.* [5], running in times $\mathcal{O}^*(4^{(r-1)k+o(rk)})$ and $\mathcal{O}^*(4^{(r-0.5)k+o(rk)})$, respectively. These algorithms, based on the randomized divide-and-conquer technique [7], also achieved the previous best running times for the weighted versions of $(r,k)$-DM and $(r,k)$-SP.

**Algorithms for $(3,k)$-DM and $(3,k)$-SP.** Many algorithms have been designed for

[I] The $\mathcal{O}^*$ notation is used to suppress polynomial factors in the running time.

the special cases of $(3,k)$-DM and $(3,k)$-SP (see Table 2). In particular, the approach in [8] uses a clever combination of dynamic programming (embedded in a color coding framework) and iterative expansion, derandomized using hash families. However, the specialized algorithms achieve neither the previous best deterministic running times for $(3,k)$-DM or $(3,k)$-SP, nor the previous best running times for the weighted versions of these problems. These are achieved by the above mentioned algorithms of Chen *et al.* [5], which, in the special cases of $(3,k)$-DM and $(3,k)$-SP, run in times $\mathcal{O}^*(16^{k+o(k)}) = \mathcal{O}^*(2.52^{3k})$ and $\mathcal{O}^*(32^{k+o(k)}) = \mathcal{O}^*(3.175^{3k})$, respectively.

| Reference | Problem | Weighted? | Algorithm | Running Time |
|---|---|---|---|---|
| Liu *et al.* [24] | SP | Yes | D | $\mathcal{O}^*(2097.152^k) = \mathcal{O}^*(12.8^{3k})$ |
| Wang *et al.* [31] | SP | Yes | D | $\mathcal{O}^*(432.082^k) = \mathcal{O}^*(7.561^{3k})$ |
| Chen *et al.* [8] | DM | No | D | $\mathcal{O}^*(21.907^k) = \mathcal{O}^*(2.799^{3k})$ |
| Liu *et al.* [25] | SP | No | D | $\mathcal{O}^*(97.973^k) = \mathcal{O}^*(4.611^{3k})$ |
| | DM | No | D | $\mathcal{O}^*(21.254^k) = \mathcal{O}^*(2.771^{3k})$ |
| | DM | No | R | $\mathcal{O}^*(12.488^k) = \mathcal{O}^*(2.321^{3k})$ |
| Wang *et al.* [32] | SP | No | D | $\mathcal{O}^*(43.615^k) = \mathcal{O}^*(3.521^{3k})$ |
| Björklund *et al.* [2] | SP | No | R | $\mathcal{O}^*(3.344^k) = \mathcal{O}^*(1.496^{3k})$ |
| | DM | No | R | $\mathcal{O}^*(2^k) = \mathcal{O}^*(1.26^{3k})$ |
| This work | SP | Yes | D | $\mathcal{O}^*(12.155^k) = \mathcal{O}^*(2.3^{3k})$ |
| | DM | Yes | D | $\mathcal{O}^*(8.125^k) = \mathcal{O}^*(2.011^{3k})$ |
| | DM | No | D | $\mathcal{O}^*(8.042^k) = \mathcal{O}^*(2.004^{3k})$ |

TABLE 2

*Algorithms for* $(3,k)$-DM *and* $(3,k)$-SP*, with D denoting deterministic algorithms and R denoting randomized algorithms.*

**Kernels for $(r,k)$-DM and $(r,k)$-SP.** Roughly speaking, we say that a problem has a kernel of size $O(f(k))$, for some function $f$, if instances of this problem can be "efficiently translated" to instances of size $O(f(k))$ (see Section 2.2). Chen et al. [5] gave kernels of size $O(r^r r k^r \log W)$ for the weighted versions of $(r,k)$-DM and $(r,k)$-SP. Fellows et al. [14] gave kernels of size $O(r!r(k-1)^r)$ for $(r,k)$-DM and $(r,k)$-SP, which can be extended to kernels of the same size (up to a factor of $\log W$) for the weighted versions of these problems. Dell et al. [10] proved that $(r,k)$-DM is unlikely to admit a kernel of size $O(f(r)k^{r-\epsilon})$ for any function $f(r)$ and $\epsilon > 0$ (improving upon a result by Hermelin et al. [20]).

**1.2. The Method of Representative Families.** The method of representative families is a general approach for developing efficient dynamic programming-based parameterized algorithms. The idea of using representative families was introduced by Monien in [27]. In this paper, we rely on a recent powerful computation of representative sets by Fomin et al. [17]. We briefly describe the basic notion of representative families here (the reader is referred to Section 2 for the details), and then discuss related results. In a dynamic programming framework, one might think of every intermediate stage of the algorithm as a location storing several partial solutions, with the hope that one of them would "lead to" a final solution. In many settings, a solution (when it exists) can be viewed as a split into two disjoint parts — where one can be thought of as the partial part that we would like to store, and the other is the part that we are hoping to encounter down the line. The inherent disjointness of

these parts suggests that perhaps all partial solutions need not be maintained, and instead, as long as there is some witness that can be evolved into a complete solution, correctness is guaranteed.

A nice illustration of this is in the dynamic programming for $k$-PATH, were paths of all intermediate lengths $1 \leq i \leq k$ are stored as we work our way up to a path of length $k$. If there is indeed a path of length $k$, say $P := \{v_1, \ldots, v_{k+1}\}$, then at the $i^{th}$ stage of the dynamic programming, it is not critical that we store the specific subpath $P_i := \{v_1, \ldots, v_i\}$. Indeed, as long as we have *some* path $\{u_1, \ldots, u_i\}$ disjoint from $P \setminus P_i$, such that $u_i v_{i+1} \in E$, we do not sacrifice correctness. Thus it is conceivable that we might be able to make progress while storing a significantly small subset of the large space of partial solutions. This is the idea exploited for the representative sets based algorithm for $k$-PATH [17].

We note that the classic color coding approach [1] is a rather clever way of capturing this notion, albeit with an element of randomization. The notion of representative families, on the other hand, formalizes this intuition in a combinatorial fashion. The definition of a representative family is as follows. Let $\mathcal{S}$ be a family of subsets of size $p$ of a universe $U$. A subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is *$q$-representative* for $\mathcal{S}$ if for every set $Y \subseteq U$ of size at most $q$, if there is a set $X \in \mathcal{S}$ disjoint from $Y$, then there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from $Y$.

For this definition to be useful, we need to know that small representative families exist. A classical result due to Bollobás indicates that every family of sets of size $p$ has a $q$-representative family with at most $\binom{p+q}{p}$ sets [3]. The next natural issue is that of computational overhead, which brings us to the question of whether these representative families can be found, and at what cost. Fortunately, the combinatorial proofs turn out to be constructive, and algorithmic versions have been established. Monien [27] computed $q$-representative families of size $\sum_{i=0}^{q} p^i$ in time $O(|\mathcal{S}|pq\sum_{i=0}^{q} p^i)$, and Marx [26] computed $q$-representative families of size $\binom{p+q}{p}$ in time $O(|\mathcal{S}|^2 p^q)$. Recently, Fomin et al. [17] introduced a powerful technique which enables us to compute $q$-representative families of size $\binom{p+q}{p}2^{o(p+q)}\log|U|$ in time $O(|\mathcal{S}| \cdot ((p+q)/q)^q \cdot 2^{o(p+q)} \log|U|)$, thus significantly improving the previous results. There are a number of problems, such as $k$-PATH [17], MULTILINEAR DETECTION [16], GRAPH MOTIF [29] and $k$-INTERNAL OUT-BRANCHING [30], that are solved using this technique. The definition of representative family can be extended to matroids and computation of representative families in linear matroids with applications are discussed in [17]. A time-size trade off in the computation of representative families were discussed in [16, 30].

**1.3. Our Contribution.** Dynamic programming is a very natural way to solve matching and packing problems. Consider the $(3,k)$-DM problem, which asks for at least $k$ mutually disjoint sets from a 3-uniform family $\mathcal{F} \subseteq U_1 \times U_2 \times U_3$. Let $n = \sum_{i=1}^{3} |U_i|$. Let $S[i]$ be the collection of all matchings of size exactly $i$ (i.e., all families that contain exactly $i$ mutually disjoint sets from $\mathcal{F}$). Note that $S[1]$ is easy to compute (indeed, $S[1]$ is simply $\mathcal{F}$). Moreover, given $S[i-1]$, $S[i]$ can be easily computed: For each matching in $S[i-1]$ and each a set in $\mathcal{F}$, if the matching and the set are disjoint, we combine them to obtain a matching that we add to $S[i]$. The above procedure is not efficient, and in fact, does not even result in a parameterized algorithm, since $S[i]$ can potentially contain $\binom{|\mathcal{F}|}{i}$ families of sets. Therefore, in this procedure, we embed computations of representative families. That is, for the

procedure to be controlled, for every $i > 1$, we compute $S[i]$ from $S[i-1]$ and then overwrite $S[i]$ with a $(3k-3i)$-representative family for $S[i]$.[II] It is not difficult to see this would be a correct algorithm, and it follows from computations similar to Theorem 11 in [17] (also described in Section 3), that the running time would turn out to be $2.851^{3k}|\mathcal{F}| \cdot n^{\mathcal{O}(1)}$.

Our focus is to improve this naive approach by more careful dynamic programming, leading to an algorithm with running time $\mathcal{O}(2.851^{(r-1)k} \cdot |\mathcal{F}| \cdot n \log^2 n)$ for $(r,k)$-DM and an algorithm with running time $\mathcal{O}(2.851^{(r-0.5501)k} \cdot |\mathcal{F}| \cdot n \log^2 n)$ for $(r,k)$-SP Our algorithms are dynamic programming based on representative families and this framework is used for many parameterized algorithms in [17]. Our approach works also for the weighted versions of these problems with only an additional factor of $\log W$ (where $W$ is the maximum weight in the input). This factor is linear in the input, and is thus essentially optimal. For $(3,k)$-DM, we are also able to apply iterative expansion as used in [8], along with the dynamic programming using representative families. The idea of iterative expansion is to focus on the "improvement" version of the question, where the input includes a matching of size $k-1$, and the question is if there is a matching of size $k$. Clearly, an efficient algorithm for this question can be run $k-1$ times (starting with $k=1$) to find a matching of size $k$. One of the reasons it is useful to have a matching of size $k$ as input is Lemma 3.4 in [8], which states if there is an improved matching, then there is one that overlaps the given matching in a substantial way. The guarantee of this overlap can be exploited suitably to attain better running times. We use this result as well, except that we incorporate the advantages in the setting of representative families. The result is an algorithm with running time $\mathcal{O}(2.0035^{3k} \cdot |\mathcal{F}| \cdot n \log^2 n)$ for $(3,k)$-DM. This complements our algorithm for the weighted version of $(3,k)$-DM, and aims to demonstrate that representative sets-based algorithms can speed-up when used in conjunction with more traditional techniques.

Using representative families, we also obtain kernels of size $O(e^r r(k-1)^r \log W)$ for the weighted versions of $(r,k)$-DM and $(r,k)$-SP. We thus improve the previous best known kernels of size $O(r!r(k-1)^r \log W)$ for these problems, given in [14].

**Organization.** In Section 2, we discuss some definitions associated with representative families. In Section 3, we describe our simplest algorithm, which solves the weighted version of $(3,k)$-DM. This algorithm is given for the sake of clarity of the paper—it simplifies the presentation of our algorithm for the weighted version of $(r,k)$-DM (in Section 4), and our algorithm for $(3,k)$-DM (in Section 5). In Section 6, we describe our algorithm for the weighted version of $(r,k)$-SP. Finally, in Section 7, we give our kernels for the weighted versions of $(r,k)$-DM and $(r,k)$-SP.

**2. Preliminaries.** In this section we state some basic definitions related to representative families and parameterized complexity, and give an overview of the tools used in this paper. We use $\omega$ to denote the matrix multiplication exponent; the current best bound is $\omega < 2.373$ [33]. We say that a family $\mathcal{S}$ of sets is a $p$-family, if each set in $\mathcal{S}$ is of size $p$. For two families of sets $\mathcal{A}$ and $\mathcal{B}$, we define

$$\mathcal{A} \bullet \mathcal{B} = \{X \cup Y \mid X \in \mathcal{A}, Y \in \mathcal{B}, X \cap Y = \emptyset\}$$

[II] Note that $S[i]$ is a collection of families of sets. When we find representative families, we would have to consider the sets associated with the families as comprising of all the elements involved in the families. It is quite straightforward to translate families of sets to set systems and vice-versa.

**2.1. Representative families.** We first give the definition of a *q-representative family* (see also [17]).

DEFINITION 2.1 (**$q$-Representative Family**, [17]). *Given a $p$-family $\mathcal{S}$ of sets over a universe $U$, we say that a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is* $q$-representative *for $\mathcal{S}$ if the following holds: for every set $Y \subseteq U$ of size at most $q$, if there is a set $X \in \mathcal{S}$ disjoint from $Y$, then there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from $Y$. If $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is $q$-representative for $\mathcal{S}$, we write $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}$. Also, we say that $\widehat{X}$ is a $q$-representative for $X$ with respect to $Y$.*

In other words, if some set in $\mathcal{S}$ can be extended to a larger set by adding $q$ new elements, then there is a set in $\widehat{\mathcal{S}}$ that can be extended by the same $q$ elements. Representative sets are known to be transitive, and this is formally stated in the following lemma.

LEMMA 2.2 ([17]). *Let $\mathcal{S}$ be a $p$-family of sets over a universe $U$. If $\mathcal{S}' \subseteq^q_{rep} \mathcal{S}$ and $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}'$, then $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}$.*

The following definition is a weighted variant of $q$-Representative Family.

DEFINITION 2.3 (**Min/Max $q$-Representative Family**, [17]). *Given a $p$-family $\mathcal{S}$ of sets over a universe $U$ and a non-negative weight function $w : \mathcal{S} \to \mathbb{N}$, we say that a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is* min $q$-representative *(*max $q$-representative*) for $\mathcal{S}$ if the following holds: for every set $Y \subseteq U$ of size at most $q$, if there is a set $X \in \mathcal{S}$ disjoint from $Y$, then there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from $Y$ with $w(\widehat{X}) \leq w(X)$ ($w(\widehat{X}) \geq w(X)$). We use $\widehat{\mathcal{S}} \subseteq^q_{minrep} \mathcal{S}$ ($\widehat{\mathcal{S}} \subseteq^q_{maxrep} \mathcal{S}$) to denote a min $q$-representative (max $q$-representative) family for $\mathcal{S}$. Also, we say that $\widehat{X}$ is a min $q$-representative (respectively, max $q$-representative) for $X$ with respect to $Y$.*

In [17], it is shown that if we are dealing with a $p$-family of sets over a universe of size $n$, then there exists a $q$-representative family whose size is a function of $p$ and $q$ alone (and, in particular, is independent of $n$). Further, such a family can be computed in time $f(p, q, |\mathcal{S}|) \cdot \log(nW)$:

THEOREM 2.4 ([17]). *There is an algorithm that given a $p$-family $\mathcal{S}$ of sets over a universe $U$ of size $n$, an integer $q$, and a non-negative weight function $w : \mathcal{S} \to \mathbb{N}$ with maximum value $W$, computes in time $\mathcal{O}(|\mathcal{S}| \cdot \binom{p+q}{q}^{\omega-1} \cdot \log(p! \cdot n^{p^2} \cdot W))$ a subfamily $\widehat{\mathcal{S}}$ such that $|\widehat{\mathcal{S}}| = \binom{p+q}{p}$ and $\widehat{\mathcal{S}} \subseteq^q_{minrep} \mathcal{S}$ ($\widehat{\mathcal{S}} \subseteq^q_{maxrep} \mathcal{S}$). Here $\omega$ is the matrix multiplication exponent.*

We use the above computation to obtain kernels for $(r, k)$-DM and $(r, k)$-SP. For our other results, we need the following computations of [17], which are faster (although they compute slightly larger representative families in comparison).

THEOREM 2.5 ([17]). *Let $\mathcal{S}$ be a $p$-family of sets over a universe of size $n$. For a given integer $q$, a $q$-representative family $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ for $\mathcal{S}$ with at most $\binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$ sets can be computed in time $\mathcal{O}(|\mathcal{S}| \cdot (\frac{p+q}{q})^q \cdot 2^{o(p+q)} \cdot \log n)$.*

THEOREM 2.6 ([17]). *There is an algorithm that given a $p$-family $\mathcal{S}$ of sets over a universe $U$ of size $n$, an integer $q$, and a non-negative weight function $w : \mathcal{S} \to \mathbb{N}$ with maximum value $W$, computes in time $\mathcal{O}(|\mathcal{S}| \cdot (\frac{p+q}{q})^q \cdot \log n + |\mathcal{S}| \cdot \log |\mathcal{S}| \cdot \log W)$ a subfamily $\widehat{\mathcal{S}}$ such that $|\widehat{\mathcal{S}}| \leq \binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$ and $\widehat{\mathcal{S}} \subseteq^q_{minrep} \mathcal{S}$ ($\widehat{\mathcal{S}} \subseteq^q_{maxrep} \mathcal{S}$).*

The following corollary will be useful in later sections.

COROLLARY 2.7. *Let $k, p', q'$ be integers. Let $\mathcal{P}$ be a family of sets of size $p'$ over a universe $U$ of size $n$, where $|\mathcal{P}| \leq \binom{k}{p'} 2^{o(k)} \log n$, and let $S \subseteq U$ of size $q'$. Let $w : \mathcal{P} \to \mathbb{N}$ be a non-negative weight function with maximum value at most $W$. Then we can compute $\widehat{\mathcal{P} \bullet \{S\}} \subseteq^{k-p'-q'}_{maxrep} \mathcal{P} \bullet \{S\}$ of size $\binom{k}{p'+q'} 2^{o(k)} \log n$ in time $2^{o(k)} \log^2 n \cdot \max_{1 \leq i \leq q'} \left\{ \binom{k}{p'+i} \cdot \left(\frac{k}{p'+i}\right)^{k-p'-i} + \binom{k}{p'+i} \cdot \log W \right\}$.*

*Proof.* Let $S = \{s_1, \ldots, s_{q'}\}$. Remove from $\mathcal{P}$ the sets that are not disjoint from $S$. First, we compute $\mathcal{P}_1 = \mathcal{P} \widehat{\bullet \{\{s_1\}\}} \subseteq^{k-p'-1}_{maxrep} \mathcal{P} \bullet \{\{s_1\}\}$ using Theorem 2.6; then, we compute $\mathcal{P}_2 = \mathcal{P}_1 \widehat{\bullet \{\{s_2\}\}} \subseteq^{k-p'-2}_{maxrep} \mathcal{P}_1 \bullet \{\{s_2\}\}$ using Theorem 2.6, and so on. We output $\mathcal{P}_{q'}$ as the $(k-p'-q')$-representative family for $\mathcal{P} \bullet \{S\}$. Using induction on $i$, we prove that for $i \in \{0, \ldots q'\}$, $\mathcal{P}_i \subseteq^{k-p'-i}_{maxrep} \mathcal{P} \bullet \{\{s_1, \ldots s_i\}\}$ where $\mathcal{P}_0 = \mathcal{P}$. The statement is trivially true for $i = 0$. Now, suppose the statement is true for all values of $j < i$. We need to show that $\mathcal{P}_i \subseteq^{k-p'-i}_{maxrep} \mathcal{P} \bullet \{\{s_1, \ldots s_i\}\}$. Let $X \in \mathcal{P} \bullet \{\{s_1, \ldots s_i\}\}$, and $Y \subseteq U$ such that $|Y| = k - p' - i$ and $X \cap Y = \emptyset$. We need to show that there exists $X^* \in \mathcal{P}_i$ such that $X^* \cap Y = \emptyset$ and $w(X^* \setminus S) \geq w(X \setminus S)$. Consider the sets $X_i = X \setminus \{s_i\}$ and $Y_i = Y \cup \{s_i\}$. Note that $X_i \in \mathcal{P} \bullet \{\{s_1, \ldots, s_{i-1}\}\}$ and $X_i \cap Y_i = \emptyset$. By induction hypothesis, $\mathcal{P}_{i-1} \subseteq^{k-p'-(i-1)}_{maxrep} \mathcal{P} \bullet \{\{s_1, \ldots s_{i-1}\}\}$, and thus there exists $X_i^* \in \mathcal{P}_{i-1}$ such that $X_i^* \cap Y_i = \emptyset$ and $w(X_i^* \setminus S) \geq w(X_i \setminus S)$. Let $X' = X_i^* \cup \{s_i\}$. Note that $X' \in \mathcal{P}_{i-1} \bullet \{\{s_i\}\}$ and $X' \cap Y = \emptyset$. Hence, there exists $X^* \in \mathcal{P}_i = \mathcal{P}_{i-1} \widehat{\bullet \{\{s_i\}\}}$ as desired.

The size of the family $\mathcal{P}_i$, for $1 \leq i \leq q'$, is bounded by $\binom{k}{p'+i} 2^{o(k)} \log n$, due to Theorem 2.6. The running time to compute $\mathcal{P}_i$ is $\mathcal{O}(|\mathcal{P}_i| \cdot (\frac{k}{p'+i})^{k-p'-i} \cdot \log n + |\mathcal{P}_i| \cdot \log |\mathcal{P}_i| \cdot \log W)$. Hence the total running time is bounded by

$$2^{o(k)} \log^2 n \max_{1 \leq i \leq q'} \left\{ \binom{k}{p'+i} \cdot \left(\frac{k}{p'+i}\right)^{k-p'-i} + \binom{k}{p'+i} \cdot \log W \right\},$$

and this concludes the proof. □

**2.2. Parameterized Complexity.** A parameterized problem is denoted by a pair $(Q, k) \subseteq \Sigma^* \times \mathbb{N}$. The first component $Q$ is a classical language, and the number $k$ is called the parameter. Such a problem is *fixed–parameter tractable* (FPT) if there exists an algorithm that decides it in time $O(f(k)n^{O(1)})$ on instances of size $n$. A *kernelization algorithm* for a parameterized problem $\Pi$ is a polynomial-time algorithm that given an instance $(Q, k)$ of $\Pi$, returns an instance $(Q', k')$ of $\Pi$ whose size is bounded by some function $f(k)$, such that $(Q, k)$ is a yes-instance if and only if $(Q', k')$ is a yes-instance. We then say that $\Pi$ has a kernel of size $f(k)$. For a detailed overview of the notions of parameterized complexity and algorithms, the reader is referred to the books [11, 15, 28].

**2.3. Matching and Packing Problems.** The $r$-DIMENSIONAL $k$-MATCHING problem is the following. Let $U$ be a universe partitioned into $r$ parts, $U_1 \uplus \cdots \uplus U_r$, and let $\mathcal{F}$ be a family of sets from $U_1 \times \cdots \times U_r$. An $r$D-Matching is a collection of mutually disjoint sets from $\mathcal{F}$. The *size* of a matching $\mathcal{M}$, denoted by $|\mathcal{M}|$, is simply the number of sets that participate in the matching. Formally, the problem is defined as follows.

$r$-Dimensional $k$-Matching ($(r,k)$-DM)
**Input:** A universe $U := U_1 \uplus \cdots \uplus U_r$, a family $\mathcal{F} \subseteq U_1 \times \cdots \times U_r$, and a positive integer $k$.
**Parameter:** $r \cdot k$
**Question:** Does $\mathcal{F}$ have a collection of at least $k$ mutually disjoint sets?

The weighted version of this problem has an additional input, namely a weight function $w$, which assigns a positive integer weight to every set in $\mathcal{F}$. Given positive integers $k$ and $W$, here the question is whether $\mathcal{F}$ has a matching of size $k$ whose weight is at least $W$. For any $\mathcal{A} \subseteq \mathcal{F}$, we use $w(\mathcal{A})$ to denote the sum $\sum_{S \in \mathcal{A}} w(S)$. We note that in the weighted setting, we do not address the question of whether there is a matching of size *at least* $k$ whose weight is at least $W$, because this problem is not monotonic — that is, there may be no matching of size $k$ with weight at least $W$, while there may be matchings of size larger than $k$ whose weight is at least $W$. Since our algorithms only check for matchings of size up to $k$, we will be concerned with the following exact version of the question.

Weighted $r$-Dimensional $k$-Matching ($(r,k)$-WDM)
**Input:** A universe $U := U_1 \uplus \cdots \uplus U_r$, a family $\mathcal{F} \subseteq U_1 \times \cdots \times U_r$, a non-negative weight function $w : \mathcal{F} \to \mathbb{N}$, and positive integers $k, W$.
**Parameter:** $r \cdot k$
**Question:** Does $\mathcal{F}$ have a collection $\mathcal{M}$ of $k$ mutually disjoint sets such that $w(\mathcal{M}) \geq W$?

The $(3,k)$-DM (respectively, $(3,k)$-WDM) problem is the restriction of $(r,k)$-DM (respectively, $(r,k)$-WDM) to the case where $r = 3$.

The packing question in this context asks for a collection of mutually disjoint sets in an $r$-uniform set system, without any further assumptions on the family. Thus, it can be thought as a more general version of $(3,k)$-WDM. Formally, this problem is defined as follows.

Weighted $r$-Set $k$-Packing ($(r,k)$-WSP)
**Input:** A universe $U$, a family $\mathcal{F}$ of subsets of size $r$ of $U$, a non-negative weight function $w : \mathcal{F} \to \mathbb{N}$, and positive integers $k, W$.
**Parameter:** $r \cdot k$
**Question:** Does $\mathcal{F}$ have a collection $\mathcal{M}$ of $k$ mutually disjoint sets such that $w(\mathcal{M}) \geq W$?

**3. An Algorithm for Weighted 3-Dimensional $k$-Matching.** This section describes an algorithm for $(3,k)$-WDM that runs in time $\mathcal{O}(2.851^{2k} \cdot |\mathcal{F}| \cdot n \log^2 n \cdot \log W)$. Later, in Section 5 we give an algorithm for $(3,k)$-DM running in time $\mathcal{O}(2.0035^{3k} \cdot |\mathcal{F}| \cdot n \log^2 n)$. The algorithm presented in this section provides a common outline for all other algorithms presented in this paper.

Let $(U_1, U_2, U_3, \mathcal{F}, w : \mathcal{F} \to \mathbb{N}, k)$ be an instance of $(3,k)$-WDM. Recall (from Section 2.3) that $\mathcal{F} \subseteq U_1 \times U_2 \times U_3$, and the problem involves finding $k$ mutually disjoint sets in $\mathcal{F}$ whose weight is at least $W$. For a set $S \in \mathcal{F}$, we let $S[i]$ denote the element in $S \cap U_i$, that is, $S[i]$ is the element of $S$ that is from $U_i$. We sometimes refer to the

**Algorithm 1:** An Outline for Matchings/Packing Algorithms

$\mathcal{L}^{(0)} \leftarrow \{\emptyset\}$
**for** $i \in \{1, 2, \ldots, n\}$ **do**
  **Compute Partial solutions** $\mathcal{L}^{(i)}$
  **Prune Partial Solutions** $\mathcal{L}^{(i)}$
**if** $\exists \mathcal{M} \in \mathcal{L}^{(n)}$ *such that* $|\mathcal{M}| = k$ *and* $w(\mathcal{M}) \geq W$ **then**
  **return** $\mathcal{M}$.
**else**
  **return** No

element $S[i]$ as the $i^{th}$ coordinate of $S$.

The improved dynamic programming approach involves iterating over the elements in $U_3$. To this end, let $U_3 := \{c_1, c_2, \ldots, c_n\}$ (for the discussion in this section, the index $i$ of each element $c_i$ is an arbitrary and fixed choice). Moreover, let $\mathcal{M} := \{S_1, S_2, \ldots, S_t\}$ be a 3D-Matching, and let $\mathcal{M}_3 := \{S_i[3] \mid i \in [t]\} \subseteq U_3$. We define the *maximum last index* of $\mathcal{M}$, denoted by $\lambda(\mathcal{M})$, as the largest index $i$ for which $c_i \in \mathcal{M}_3$. For the empty matching $\emptyset$, $\lambda(\emptyset) = 0$. In the $i^{th}$ iteration of our algorithm, we would like to store all matchings whose maximum last index is at most $i$. However, it turns out that storing all possible matchings at every stage is exponentially expensive — we potentially run into storing $O(n^{2i})$ matchings at the $i^{th}$ step. In [8], this problem (for the unweighted case) is mitigated using color coding. We will now discuss how we can use the notion of max representative families instead, which has the dual advantage of being faster and deterministic.

We are now ready to describe our first algorithm, Algorithm 1, whose outline is given below. The heart of Algorithm 1 consists of two modules – **Compute Partial Solutions** $\mathcal{L}^{(i)}$ and **Prune Partial Solutions** $\mathcal{L}^{(i)}$. In $\mathcal{L}^{(i)}$, we store carefully chosen 3D-Matchings whose maximum last index is at most $i$. In the module **Compute Partial Solutions** $\mathcal{L}^{(i)}$, we compute $\mathcal{L}^{(i)}$ by considering each matching $\mathcal{M}$ of $\mathcal{L}^{(i-1)}$, and checking if it can be extended to a matching whose maximum last index is at most $i$. For a fixed $\mathcal{M}$, this check is performed by iterating over all elements $S \in \mathcal{F}$ such that $S[3] = c_i$ and extending $\mathcal{M}$ to $\mathcal{M} \cup S$ whenever $\mathcal{M} \cap S = \emptyset$. Formally,

$$\mathcal{L}^{(i)} \leftarrow \{\mathcal{M} \cup S \mid \mathcal{M} \in \mathcal{L}^{(i-1)}, S \in \mathcal{F}, \mathcal{M} \cap S = \emptyset, S[3] = c_i\} \cup \mathcal{L}^{(i-1)} \tag{3.1}$$

The pseudocode of the second module, **Prune Partial Solutions** $\mathcal{L}^{(i)}$, is given in Algorithm 2. In this module, we compute a representative family for $\mathcal{L}^{(i)}$. Since $\mathcal{L}^{(i)}$ is a collection of matchings of varying sizes, we will need an appropriate version of $\mathcal{L}^{(i)}$ that is suitable for Theorem 2.6. To this end, we first partition the set $\mathcal{L}^{(i)}$ — the part denoted by $\mathcal{L}^{(i)}_j$ contains all matchings from $\mathcal{L}^{(i)}$ of size $j$. Note that this is simply done to ensure uniformity of size.

To discuss the rest of the module, we define some notations. We let $\mathcal{Q}^{(i)} := \{\mathcal{M} \mid \lambda(\mathcal{M}) \leq i\}$ and $\mathcal{Q}^{(i)}_j := \{\mathcal{M} \mid \mathcal{M} \in \mathcal{Q}^{(i)}, |\mathcal{M}| = j\}$. Notice that the $\mathcal{Q}^{(i)}_j$'s constitute a partition of the set $\mathcal{Q}^{(i)}$ based on matching size; in other words, $\mathcal{Q}^{(i)} := \bigcup_{j=0}^{n} \mathcal{Q}^{(i)}_j$. Recall (from Section 1.3) that a naive application of the method of representative families would lead to a running time of $\mathcal{O}^*(2.851^{3k})$. To use the representative

**Algorithm 2:** Prune Partial Solutions $\mathcal{L}^{(i)}$

$\mathcal{R}_0^{(0)} \leftarrow \{\emptyset\}$
$\mathcal{R}_j^{(0)} \leftarrow \emptyset$, for all $1 \leq j \leq k$
**for** $j \in \{0, 1, \ldots, k\}$ **do**
  $\mathcal{L}_j^{(i)} \leftarrow \{\mathcal{M} \mid \mathcal{M} \in \mathcal{L}^{(i)}, |\mathcal{M}| = j\}$
  $\mathcal{P}_j^{(i)} \leftarrow \{\mathcal{M}_{12} \mid \mathcal{M} \in \mathcal{L}_j^{(i)}\}$
  $\gamma \leftarrow \{(\mathcal{M}, \mathcal{M}_{12}) \mid \mathcal{M}$ is a matching in $\mathcal{L}_j^{(i)}\}$
  $w' \leftarrow \{(\mathcal{M}_{12}, w(\widetilde{\gamma}(\mathcal{M}_{12}))) \mid \mathcal{M}_{12} \in \mathcal{P}_j^{(i)}\}$
  Compute $\mathcal{R}_j^{(i)} \subseteq^{2(k-j)}_{maxrep} \mathcal{P}_j^{(i)}$ using Theorem 2.6
  $\mathcal{L}_j^{(i)} \leftarrow \{\widetilde{\gamma}(\mathcal{M}_{12}) \mid \mathcal{M}_{12} \in \mathcal{R}_j^{(i)}\}$
$\mathcal{L}^{(i)} \leftarrow \bigcup_{j=0}^{k} \mathcal{L}_j^{(i)}$

families more efficiently, we would like to split the elements of the matchings into two parts. We will first collect the parts of the matching that come from $(U_1 \cup U_2)$, and then store separately a map that completes the first part to the complete matching. This will allow us to apply the dynamic programming approach suggested above. To this end, for a matching $\mathcal{M}$, we let $\mathcal{M}_{12}$ denote the subset of $(U_1 \cup U_2)$ obtained by projecting the elements of $\mathcal{M}$ on their first two coordinates, that is, $\mathcal{M}_{12} := \bigcup_{S \in \mathcal{M}} \{S[1], S[2]\}$. On the other hand, let

$$\gamma := \{(\mathcal{M}, \mathcal{M}_{12}) \mid \mathcal{M} \text{ is a matching in } \mathcal{F}\}.$$

In $\gamma$, we are merely storing the associations of $\mathcal{M}_{12}$ with the matchings that they "came from". Observe that $\gamma$ might (by definition) contain multiple entries with the same second coordinate. On the other hand, when the algorithm stores the associations in $\gamma$, we will see that it is enough to maintain one maximum weighted entry for each $\mathcal{M}_{12}$. To this end, we define the function $\widetilde{\gamma}$ as follows. Let $\preceq$ be an arbitrary total order on the set of all matchings in $\mathcal{F}$. For a set $S \subseteq U_1 \cup U_2$, we define $\widetilde{\gamma}(S)$ as the smallest matching $\mathcal{M}$ (with respect to $\preceq$) among the maximum weighted matchings in the set $\{\mathcal{M}' \mid (\mathcal{M}', S) \in \gamma\}$. If $\widetilde{\gamma}(S)$ is $\mathcal{M}$, then we say that $\mathcal{M}$ is the matching associated with $S$. Finally, for a set $S$ and a matching $\mathcal{M}$, we abuse notation and say that $\mathcal{M}$ is disjoint from $S$ (notationally, $\mathcal{M} \cap S = \emptyset$) to mean that $T \cap S = \emptyset$ for all $T \in \mathcal{M}$.

Next, we associate with every matching $\mathcal{M} \in \mathcal{L}_j^{(i)}$, a set that consists of the first two indices of every set in $\mathcal{M}$. Recall that this is denoted by $\mathcal{M}_{12}$. The collection of sets that correspond to matchings in $\mathcal{L}_j^{(i)}$ is denoted by $\mathcal{P}_j^{(i)}$. We use $\gamma$ to store the associations between the sets and the original matchings. Note that $\gamma$ might have multiple pairs with the same second index and the same weight $w$ for the first index, but this will be irrelevant (it would simply mean that $\mathcal{M}_{12}$ can be "pulled back" to multiple matchings, each of which would be equally valid). We then define a weight function $w' : \mathcal{P}_j^{(i)} \to \mathbb{N}$. For all $\mathcal{M}_{12} \in \mathcal{P}_j^{(i)}$, we set $w'(\mathcal{M}_{12}) = w(\widetilde{\gamma}(\mathcal{M}_{12}))$. Now, the central step of the algorithm is to compute a max $2(k-j)$-representative family $\mathcal{R}_j^{(i)}$ for $\mathcal{P}_j^{(i)}$. Once we have the representative family, we revise $\mathcal{L}_j^{(i)}$ to only include the matchings associated with the sets in $\mathcal{R}_j^{(i)}$.

The correctness of the algorithm relies on the fact that at each step, instead of the complete family of partial solutions $\mathcal{Q}^{(i)}$, it suffices to store only a representative family for $\mathcal{Q}^{(i)}$. Also, we will show that the family computed by the algorithm, $\mathcal{L}^{(i)}$, is indeed a representative family for $\mathcal{Q}^{(i)}$. The analysis of the running time will be quite straightforward in the light of Theorem 2.6, and we will discuss it after arguing the correctness.

Let $\mathcal{X}_j^{(i)} := \{\mathcal{M}_{12} \mid \mathcal{M} \in \mathcal{Q}_j^{(i)}\}$. We assign a weight function $w' : \mathcal{X}_j^{(i)} \to \mathbb{N}$ as follows. For every $\mathcal{M}_{12} \in \mathcal{X}_j^{(i)}$, $w'(\mathcal{M}_{12}) = \max\{w(\mathcal{M}^\star) \mid \mathcal{M}^\star \in \mathcal{Q}_j^{(i)}, \mathcal{M}_{12} = \mathcal{M}_{12}^\star\}$. Our first claim is the following.

LEMMA 3.1. *For all $0 \leq i \leq n$ and $0 \leq j \leq k$, the set $\mathcal{R}_j^{(i)}$ is a max $2(k-j)$-representative family for $\mathcal{X}_j^{(i)}$.*

*Proof.* To prove this lemma, we need to show that for all $Y \subseteq U_1 \cup U_2$ such that $|Y| \leq 2(k-j)$, if there exists a set $Z \in \mathcal{X}_j^{(i)}$ such that $Y \cap Z = \emptyset$, then there also exists $Z^\star \in \mathcal{R}_j^{(i)}$ such that $Z^\star \cap Y = \emptyset$ and $w'(Z^\star) \geq w'(Z)$. Recall that in this situation, we say that $Z^\star$ is a max $2(k-j)$-representative for $Z$ with respect to $Y$. The proof is by induction on $i$. The base case is when $i = 0$. Observe that:

$$\mathcal{R}_j^{(0)} = \mathcal{X}_j^{(0)} = \begin{cases} \{\emptyset\} & \text{if } j = 0 \\ \emptyset & \text{Otherwise} \end{cases}$$

Hence $\mathcal{R}_j^{(0)}$ is $2(k-j)$-representative family for $\mathcal{X}_j^{(0)}$ for all $0 \leq j \leq k$. The induction hypothesis states that $\mathcal{R}_j^{(i)}$ is a max $2(k-j)$-representative family for $\mathcal{X}_j^{(i)}$, for all $0 \leq j \leq k$. We will now show that $\mathcal{R}_j^{(i+1)}$ is a max $2(k-j)$-representative family for $\mathcal{X}_j^{(i+1)}$, for all $0 \leq j \leq k$. Note that the case when $j = 0$ is easily handled, since $\emptyset \in \mathcal{R}_0^{(i+1)}$. For the rest of this proof we assume that $j \in \{1, \ldots, k\}$. Let $Y \subseteq U_1 \cup U_2$ such that $|Y| \leq 2(k-j)$, and suppose there exists a set $Z \in \mathcal{X}_j^{(i+1)}$ such that $Z \cap Y = \emptyset$. Let $\mathcal{M}^Z$ be the matching associated with $Z$ and hence $w'(Z) = w(\mathcal{M}^Z)$. Since $\mathcal{M}^Z$ is derived from an element of $\mathcal{X}_j^{(i+1)}$, note that $\lambda(\mathcal{M}^Z) \leq i+1$. We distinguish two cases, depending on whether $\mathcal{M}^Z$ contains a set with $c_{i+1}$ as the third coordinate.

**Case 1.** $\mathcal{M}^Z$ contains a set with $c_{i+1}$ as the third coordinate (see also Figure 1).

Let $S \in \mathcal{M}^Z$ be such that $c_{i+1} \in S$. Define the smaller matching $\mathcal{M}^{Z \setminus S} := \mathcal{M}^Z \setminus S$. Note that $|\mathcal{M}^{Z \setminus S}| = j - 1$ and $\lambda(\mathcal{M}^{Z \setminus S}) \leq i$. Hence, $\mathcal{M}^{Z \setminus S} \in \mathcal{Q}_{j-1}^{(i)}$ and $\mathcal{M}_{12}^{Z \setminus S} \in \mathcal{X}_{j-1}^{(i)}$. Let $A = \mathcal{M}_{12}^{Z \setminus S}$. By the definition of $w'$, $w'(A) \geq w(\mathcal{M}^{Z \setminus S})$. Now consider the set $Y^S = Y \cup \{S[1], S[2]\}$. Note that $|Y^S| \leq 2(k-j+1)$, since $|Y| \leq 2(k-j)$. It is also easy to check that $A \cap Y^S = \emptyset$. By the induction hypothesis, we have that $\mathcal{R}_{j-1}^{(i)}$ contains a $2(k-j+1)$-representative of $A$ with respect to $Y^S$. Let us denote this representative by $B$. Note that $B \cap Y^S = \emptyset$ and $w'(B) \geq w'(A)$ by definition. Let $\mathcal{M}^B$ be the matching associated with $B$. Since $B \in \mathcal{R}_{j-1}^{(i)}$, we have that $\mathcal{M}^B \in \mathcal{L}_{j-1}^{(i)}$, and in particular, the maximum last index of $\mathcal{M}^B$ is at most $i$, and $w(\mathcal{M}^B) = w'(B)$. Since the first two elements of $S$ are disjoint from $B$ and the last element of $S$ is $c_{i+1}$, we have that $S$ is disjoint from $\mathcal{M}^B$. Thus, $\mathcal{M}^B \cup S$ is a matching of size $j$ with

maximum last index $(i+1)$. Let us denote this matching by $\mathcal{M}^{B|S}$. Note that

$$\begin{aligned} w(\mathcal{M}^{B|S}) &= w(\mathcal{M}^B) + w(S) \\ &\geq w(\mathcal{M}^{Z\setminus S}) + w(S) \quad (\text{Since } w(\mathcal{M}^B) = w'(B) \geq w'(A) \geq w(\mathcal{M}^{Z\setminus S}) \\ &= w'(Z) \qquad\qquad (\text{Since } w(\mathcal{M}^{Z\setminus S}) + w(S) = w(\mathcal{M}^Z) = w'(Z)) \end{aligned}$$

It is easy to check that $\mathcal{M}^{B|S}$ is present in $\mathcal{L}_j^{(i+1)}$, and correspondingly, $\mathcal{M}_{12}^{B|S}$ is present in $\mathcal{P}_j^{(i+1)}$, and $w'(\mathcal{M}_{12}^{B|S}) \geq w'(Z)$. Let us use $V$ to denote the set $\mathcal{M}_{12}^{B|S}$. It is immediate that $V \cap Y = \emptyset$. Since $\mathcal{R}_j^{(i+1)}$ is a max representative family for $\mathcal{P}_j^{(i+1)}$, it contains a max $2(k-j)$-representative for $V$ with respect to $Y$. We let $Z^\star$ denote this representative. Note that $Z^\star \in \mathcal{R}_j^{(i+1)}$, $w'(Z^\star) \geq w'(V) \geq w'(Z)$ and $Z^\star$ is disjoint from $Y$, which is exactly what was desired.

**Case 2.** $\mathcal{M}^Z$ contains no set with $c_{i+1}$ as the third coordinate.

In this case, we have that $\lambda(\mathcal{M}^Z) \leq i$. Clearly, $\mathcal{M}^Z$ is contained in $\mathcal{Q}_j^{(i)}$ and consequently, $\mathcal{M}_{12}^Z \in \mathcal{X}_j^{(i)}$. By the induction hypothesis, let $B$ be the max $2(k-j)$-representative for $\mathcal{M}_{12}^Z$ in $\mathcal{R}_j^{(i)}$ with respect to $Y$. Note that $B$ is disjoint from $Y$, and $w'(B) \geq w'(\mathcal{M}_{12}^Z) \geq w(\mathcal{M}^Z) = w'(Z)$. The matching associated with $B$, say $\mathcal{M}^B$, belongs to $\mathcal{L}_j^{(i)}$, and $w(\mathcal{M}^B) = w'(B)$. Further, since $\mathcal{L}_j^{(i+1)} \supseteq \mathcal{L}_j^{(i)}$, we have that $\mathcal{M}^B$ is also present in $\mathcal{L}_j^{(i+1)}$. As before, this implies that $\mathcal{M}_{12}^B \in \mathcal{P}_j^{(i+1)}$. Note that this may be equivalently stated as $B \in \mathcal{P}_j^{(i+1)}$. Now, letting $V$ denote $B$, the rest of the proof is identical to the last part of the previous case. □

Observe that any solution constructed by Algorithm 1 is always a valid matching. Therefore, if there is no matching of size $k$, Algorithm 1 always returns No. On the other hand, if given a Yes-instance, we now show that Algorithm 1 always finds a 3D-Matching of size $k$ with weight at least $W$.

Lemma 3.2. *Let $(U_1, U_2, U_3, \mathcal{F}, w : \mathcal{F} \to \mathbb{N}, k, W)$ be a* Yes*-instance of* $(3,k)$-WDM. *Then, Algorithm 1 successfully computes a 3D-Matching of size $k$ with weight at least $W$.*

*Proof.* Let $(U_1, U_2, U_3, \mathcal{F}, w : \mathcal{F} \to \mathbb{N}, k, W)$ be a Yes-instance of $(3,k)$-WDM. Let $\mathcal{M}$ be a $k$-sized 3D-Matching in $\mathcal{F}$ such that $w(\mathcal{M}) \geq W$. Recall that $\mathcal{Q}_j^{(i)}$ is the set of all 3D-Matchings of size $j$ with maximum last index at most $i$, and $\mathcal{X}_j^{(i)}$ contains the projections of these matchings on their first two coordinates. Therefore, $\mathcal{M} \in \mathcal{Q}_k^{(n)}$ and $\mathcal{M}_{12} \in \mathcal{X}_k^{(n)}$. By Lemma 3.1, we have that $\mathcal{R}_k^{(n)}$ is a max 0-representative family for $\mathcal{X}_k^{(n)}$. Since $\mathcal{R}_k^{(n)} \subseteq_{maxrep}^0 \mathcal{X}_k^{(n)}$, we have that $\mathcal{R}_k^{(n)}$ contains a 0-representative $Z^\star$ for $\mathcal{M}_{12}$ with respect to $\emptyset$. So we have $w'(Z^\star) \geq w'(\mathcal{M}_{12})$. Let $\mathcal{M}^\star$ be the matching associated with $Z^\star$. Therefore $w(\mathcal{M}^\star) = w'(Z^\star) \geq w'(\mathcal{M}_{12}) = w(\mathcal{M})$. Therefore, in Step 10 of Algorithm 2, we have that $\mathcal{L}^{(n)}$ contains the matching $\mathcal{M}^\star$, which is then returned as output in Step 6 of Algorithm 1. □

Lemma 3.3. *The running time of Algorithm 1 is bounded by* $\mathcal{O}(2.851^{2k} \cdot |\mathcal{F}| \cdot n \log^2 n \cdot \log W)$.

*Proof.* Let $1 \leq i \leq n$ and $1 \leq j \leq k$ be fixed, and let us consider the running time of Step 8 of Algorithm 2, which is the computation of a max $2(k-j)$-representative family. By Theorem 2.6, the running time of this computation is bounded by:

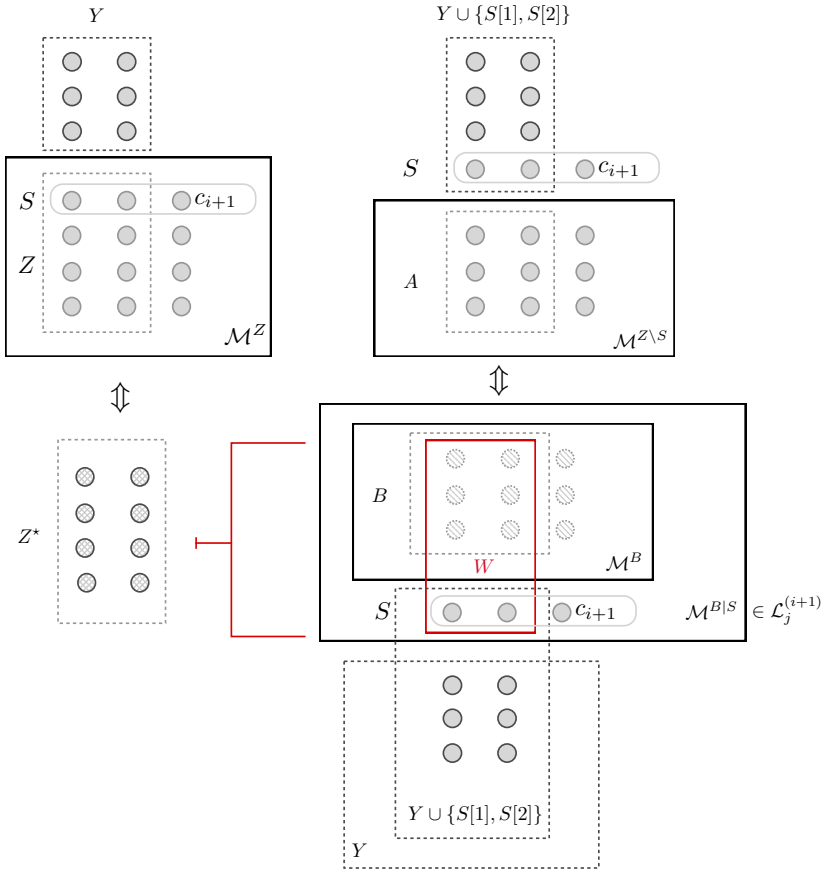


FIG. 1. *A schematic view of the proof of Lemma 3.1. It is meant to be read in clockwise order. Note that $B$ is a representative for $A$ with respect to $Y \cup \{S[1], S[2]\}$, and $Z^\star$ is a representative for $B \cup S$ with respect to $Y$. This eventually implies that $Z^\star$ is a representative for $Z$ with respect to $Y$.*

$$\mathcal{O}\left(\left(\frac{2k}{2(k-j)}\right)^{2(k-j)} \cdot 2^{o(k)} \cdot |\mathcal{P}_j^{(i)}| \cdot \log n + |\mathcal{P}_j^{(i)}| \cdot \log(|\mathcal{P}_j^{(i)}|) \cdot \log W\right).$$

Note that the size of the family $\mathcal{P}_j^{(i)}$ is given by the size of the representative families computed in the previous iteration: first, those that involved matchings of size $j$, and second, those that involved matchings of size $(j-1)$ and were updated to matchings of size $j$ in the first module. Therefore, $|\mathcal{P}_j^{(i)}| \leq |\mathcal{R}_j^{(i-1)}| + |\mathcal{F}| \cdot |\mathcal{R}_{j-1}^{(i-1)}|$. Again, by Theorem 2.6, the sizes of the representative families are bounded, thus:

$$|\mathcal{P}_j^{(i)}| \leq \left(\binom{2k}{2j} + \binom{2k}{2j-2}|\mathcal{F}|\right) \cdot 2^{o(k)} \cdot \log n.$$

Noting that Algorithm 2 runs at most $n$ times and the **for** loop in Algorithm 2 runs at most $k$ times, the overall running time can be bounded by:

$$kn \cdot \max_{1 \leq j \leq k} \left(\binom{2k}{2j} \cdot \left(\frac{2k}{2(k-j)}\right)^{2(k-j)}\right) \cdot 2^{o(k)} \cdot |\mathcal{F}| \cdot \log^2 n \cdot \log W.$$

The maximum is achieved at $2j = \alpha \cdot 2k$, where $\alpha = 1 + \dfrac{1-\sqrt{1+4e}}{2e}$. Therefore, the running time above simplifies to $\mathcal{O}(2.851^{2k} \cdot |\mathcal{F}| \cdot n \log^2 n \cdot \log W)$. □

**4. An Algorithm for Weighted $r$-Dimensional Matching.** In this section, we describe how to generalize Algorithm 1 (given in Section 3) to solve $(r,k)$-WDM in time $\mathcal{O}(2.851^{(r-1)k} \cdot |\mathcal{F}| \cdot n \log^2 n \cdot \log W)$. First, we easily modify the notation to correspond to the more general dimension $r$ as follows. Let $U_r = \{c_1, c_2, \ldots, c_n\}$. For a set $S \in \mathcal{F}$ and an $r$D-Matching $\mathcal{M}$, let $S_{[r-1]} = \{S[i] \mid 1 \leq i \leq r-1\}$ and $\mathcal{M}_{[r-1]} := \bigcup_{S \in \mathcal{M}} S_{[r-1]}$. We define the maximum last index of $\mathcal{M}$ as $\lambda(\mathcal{M}) := \max\{i \mid S \in \mathcal{M}, S \cap U_r = c_i\}$. Correspondingly,

$$\gamma_j := \{(\mathcal{M}, \mathcal{M}_{[r-1]}) \mid \mathcal{M} \text{ is a matching of size } j \text{ in } \mathcal{F}\}.$$

Now, we define $\mathcal{Q}^{(i)}$, $\mathcal{Q}_j^{(i)}$ and $\widetilde{\gamma}_j(S)$ exactly as before. Let $\mathcal{X}_j^{(i)} := \{\mathcal{M}_{[r-1]} \mid \mathcal{M} \in \mathcal{Q}_j^{(i)}\}$. We define a weight function $w' : \mathcal{X}_j^{(i)} \to \mathbb{N}$ as follows. For every $\mathcal{M}_{[r-1]} \in \mathcal{X}_j^{(i)}$, $w'(\mathcal{M}_{[r-1]}) = \max\{w(\mathcal{M}^\star) \mid \mathcal{M}^\star \in \mathcal{Q}_j^{(i)}, \mathcal{M}_{[r-1]} = \mathcal{M}^\star_{[r-1]}\}$.

Using this notation, the algorithm for $(r,k)$-WDM, excluding the module **Compute Partial Solutions** $\mathcal{L}^{(i)}$, is a straightforward generalization of Algorithm 1. By $r$D-Matching Algorithm, we refer to the new algorithm.

First, we explain why we need to replace the previous **Compute Partial Solutions** $\mathcal{L}^{(i)}$ module. Using the previous module, we get that for all $1 \leq i \leq n$ and $1 \leq j \leq (k-1)$, $|\mathcal{P}_j^{(i)}| \leq |\mathcal{R}_j^{(i-1)}| + |\mathcal{F}| \cdot |\mathcal{R}_{j-1}^{(i-1)}|$ (see the proof of Lemma 3.3). Now, we can bound $|\mathcal{R}_{j-1}^{(i-1)}|$ by $\binom{(r-1)k}{(r-1)(j-1)} \cdot 2^{o(rk)} \cdot \log n$, by Theorem 2.6, which is not necessarily bounded by $\binom{(r-1)k}{(r-1)j} \cdot 2^{o(rk)} \cdot \log n$, since $r$ might not be a constant. Thus, we cannot obtain a running time of $\mathcal{O}^*(2.851^{(r-1)k})$ by a straightforward generalization of the proof of Lemma 3.3. In other words, the difficulty stems from the fact that we use $(r-1)(k-(j-1))$-representative families to compute $(r-1)(k-j)$-representative families. We resolve this issue by using Corollary 2.7.

The new **Compute Partial Solutions** $\mathcal{L}^{(i)}$ module, whose pseudocode is given in Algorithm 3, uses Corollary 2.7 to perform an initial pruning of $\mathcal{L}^{(i)}$ (while computing it from $\mathcal{L}^{(i-1)}$ and $\mathcal{F}$). In Algorithm 3, the collection of sets corresponding to matchings in $\mathcal{L}_j^{(i-1)}$ are stored in $\mathcal{R}_j^{(i-1)}$. By computing $\mathcal{R}_j(S) \subseteq^{(r-1)(k-j)}_{maxrep} \mathcal{R}_{j-1}^{(i-1)} \bullet \{S_{[r-1]}\}$

**Algorithm 3:** Compute Partial Solutions $\mathcal{L}^{(i)}$

$\gamma_0 \leftarrow \{(\{\emptyset\}, \{\emptyset\})\}$

$\mathcal{L}_j^{(i-1)} \leftarrow \{\mathcal{M} \mid \mathcal{M} \in \mathcal{L}^{(i-1)}, |\mathcal{M}| = j\}$

$\mathcal{R}_j^{(i-1)} \leftarrow \{\mathcal{M}_{[r-1]} \mid \mathcal{M} \in \mathcal{L}_j^{(i-1)}\}$

**for** $j \in \{1, \ldots, k\}$, $S \in \mathcal{F}$ *such that* $S[r] = c_i$ **do**

  Compute $\mathcal{R}_j(S) \subseteq_{maxrep}^{(r-1)(k-j)} \mathcal{R}_{j-1}^{(i-1)} \bullet \{S_{[r-1]}\}$ using Corollary 2.7

$\mathcal{L}^{(i)} \leftarrow \bigcup_{j=1}^{k} \left(\{\widetilde{\gamma}_j(\mathcal{M}_{[r-1]}) \cup S \mid \mathcal{M}_{[r-1]} \cup S_{[r-1]} \in \mathcal{R}_j(S), S[r] = c_i\}\right) \cup \mathcal{L}^{(i-1)}$

for any set $S$ such that $S[r] = c_i$, we are computing a representative family for $\mathcal{X}_j^{(i-1)} \bullet \{S\}$; then, in the last step, the matchings are recovered from $\mathcal{R}_j(S)$ via the map $\widetilde{\gamma}_j$.

We now adapt of Lemma 3.1 to the context of the $r$D-Matching Algorithm.

Lemma 4.1. *For all $0 \leq i \leq n$, and $0 \leq j \leq k$, the set $\mathcal{R}_j^{(i)}$ is a max $((r-1)(k-j))$-representative family for $\mathcal{X}_j^{(i)}$.*

*Proof.* The proof of the lemma is a straightforward generalization of the proof of Lemma 3.1, excluding the last paragraph of Case 1—this is the location where the new **Compute Partial Solutions** $\mathcal{L}^{(i)}$ module effects the proof. Now, due to the pruning in this module, we cannot claim that $\mathcal{M}^{B|S}$ is present in $\mathcal{L}_j^{(i+1)}$, but only that it is present in $\{\mathcal{M} \cup S \mid \mathcal{M} \in \mathcal{L}^{(i-1)}, S \in \mathcal{F}, \mathcal{M} \cap S = \emptyset, S[r] = c_i\}$. Clearly, for the rest of the proof to hold (with straightforward modifications), it is sufficient to show that there is $\widetilde{\mathcal{M}} \in \mathcal{L}_j^{(i+1)}$ such that $w(\widetilde{\mathcal{M}}) \geq w(\mathcal{M}^{B|S})$. By Corollary 2.7, we have that $\bigcup_{S \in \mathcal{F} \text{ s.t. } S[i]=c_i} \mathcal{R}_j(S) \subseteq_{maxrep}^{(r-1)(k-j)} \mathcal{A}$, where $\mathcal{A} = \{\mathcal{M}_{[r-1]} \cup S \setminus \{c_i\} \mid \mathcal{M} \in \mathcal{L}_{j-1}^{(i)}, S \in \mathcal{F}, S[r] = c_i, S_{[r-1]} \cap M_{[r-1]} = \emptyset\}$. Observe that $\mathcal{M}_{[r-1]}^{B|S} \in \mathcal{A}$, and therefore there exists $C \in \mathcal{A}$ such that $w'(C) \geq w'(B \cup S \setminus \{c_i\}) \geq w(\mathcal{M}^{B|S})$. Thus, we can choose $\widetilde{\mathcal{M}}$ as the matching associated with $C$. □

Using arguments very similar to those given in the proof of Lemma 3.2, we can prove the following lemma.

Lemma 4.2. *Let $(U_1, U_2, \ldots, U_r, \mathcal{F}, w : \mathcal{F} \to \mathbb{N}, k, W)$ be a* Yes*-instance of $(r,k)$-*WDM*. Then, $r$*D-Matching Algorithm *successfully computes an $r$D-Matching of size $k$ with weight at least $W$.*

Next, we show that the running time of $r$D-Matching Algorithm is indeed bounded by $\mathcal{O}(2.851^{(r-1)k} \cdot |\mathcal{F}| \cdot n \log^2 n \cdot \log W)$.

Lemma 4.3. *The running time of $r$*D-Matching Algorithm *is bounded by $\mathcal{O}(2.851^{(r-1)k} \cdot |\mathcal{F}| \cdot n \log^2 n \cdot \log W)$.*

*Proof.* Let $1 \leq i \leq n$ and $1 \leq j \leq k$ be fixed. Let $j' = j - 1$. First, consider the

running time of Step 5 in Algorithm 3. Due to Corollary 2.7, it is bounded by:

$$2^{o(k)}\log^2 n \max_{1\le j''<r}\left\{\binom{(r-1)k}{(r-1)j'+j''}\cdot\left(\frac{(r-1)k}{(r-1)j'+j''}\right)^{(r-1)(k-j')-j''} + \binom{(r-1)k}{(r-1)j'+j''}\cdot\log W\right\}$$

By Corollary 2.7, $|\mathcal{P}_j^{(i)}| \le (|\mathcal{F}|+1)\cdot\binom{(r-1)k}{(r-1)j}2^{o(rk)}\log n$. Thus, by Theorem 2.6, the running time of the step generalizing Step 8 of Algorithm 2 is bounded by:

$$\mathcal{O}\left(\left(\frac{(r-1)k}{(r-1)(k-j)}\right)^{(r-1)(k-j)}\cdot 2^{o(rk)}\cdot\binom{(r-1)k}{(r-1)j}\cdot|\mathcal{F}|\cdot\log^2 n\cdot\log W\right).$$

The maximum is achieved at $(r-1)j = \alpha\cdot(r-1)k$, where $\alpha = 1+\dfrac{1-\sqrt{1+4e}}{2e}$. Therefore, the running time above simplifies to $\mathcal{O}(2.851^{(r-1)k}\cdot|\mathcal{F}|\cdot n\log^2 n\cdot\log W)$. □

We have thus obtained the following theorem.

THEOREM 4.4. *$(r,k)$-WDM can be solved in $\mathcal{O}(2.851^{(r-1)k}\cdot|\mathcal{F}|\cdot n\log^2 n\cdot\log W)$ time.*

**5. An Improved Algorithm for 3-Dimensional $k$-Matching.** In this section, we provide a faster algorithm for $(3,k)$-DM using iterative expansion. The idea is to first solve the following improvement problem: given a 3D-Matching of size $j-1$ as input, can we find a 3D-Matching of size $j$? Once we have an algorithm that solves the improvement question, we can use it as a subroutine $k-1$ times to find a matching of size $k$, by starting with a trivial matching of size one. Therefore, for the rest of this section, we focus on the improvement problem:

3D-MATCHING IMPROVEMENT
**Input:** A universe $U := U_1 \uplus U_2 \uplus U_3$, a family $\mathcal{F} \subseteq U_1\times U_2\times U_3$, and $\mathcal{K}\subseteq\mathcal{F}$, a 3D-Matching of size $k-1$.
**Parameter:** $k$
**Question:** Does $\mathcal{F}$ have a 3D-Matching of size $k$?

Let $\mathcal{M}$ be a 3D-Matching given by $\{S_1,\ldots,S_t\}$. Generalizing the notation in the previous section, we let $\mathcal{M}_{pq}$ denote the subset of $(U_p\cup U_q)$ obtained by projecting the elements of $\mathcal{M}$ on the $p$ and $q$ coordinates, that is, $\mathcal{M}_{pq} := \bigcup_{S\in\mathcal{M}}\{S[p],S[q]\}$. To use the method of iterative expansion to our advantage, we invoke the following result from [8], which states that if there is *some* matching of size $k$, then there is also one that has a large intersection with the given matching $\mathcal{K}$.

LEMMA 5.1 (Lemma 3.4 and Theorem 3.6, [8]). *Let $(U_1,U_2,U_3,\mathcal{F},\mathcal{K})$ be a* YES*-instance of* 3D-MATCHING IMPROVEMENT*, that is, $\mathcal{F}$ admits a matching $\mathcal{M}$ of size $k$.*

*Then, there is also a matching $\mathcal{M}^\star$ of size $k$ such that there exist two indices $p, q \in \{1,2,3\}$, for which $|\mathcal{M}_{pq} \cap \mathcal{K}_{pq}| \geq (4/3)(k-1)$.*

The improved algorithm for solving 3D-Matching Improvement is along the lines of Algorithm 1. The difference lies in how we compute the representative families, and this is also what brings about the improved running time.

We begin by considering the given matching, $\mathcal{K}$. By Lemma 5.1, there is always a solution that has a large common intersection with $\mathcal{K}$ (if a solution exists). In particular, if we denote this solution by $\mathcal{M}^\star$, then Lemma 5.1 indicates that there is a choice of coordinates $p, q \in \{1,2,3\}$ for which the number of elements in $\mathcal{K}_{pq} \cap \mathcal{M}^\star_{pq}$ is at least $(4/3)(k-1)$; so our first step involves guessing the coordinates $p$ and $q$, and renaming $p$ and $q$ to be 1 and 2, respectively. We also rename the remaining coordinate, $r \in \{1,2,3\} \setminus \{p,q\}$, to be 3. Thus, from now on, we assume that if the given input $(U_1, U_2, U_3, \mathcal{F}, \mathcal{K})$ is an Yes-instance then $|\mathcal{K}_{12} \cap \mathcal{M}^\star_{12}|$ is at least $(4/3)(k-1)$.

Now, our algorithm is same as Algorithm 1, apart from the module **Prune Partial Solutions** $\mathcal{L}^{(i)}$. By 3-DM Algorithm, we refer to the new algorithm. For the pseudo code of the module **Prune Partial Solutions** $\mathcal{L}^{(i)}$ see Algorithm 4. In the new module, we first guess the intersection $\mathcal{K}_{12} \cap \mathcal{M}^\star_{12}$, by iterating over all the subsets of $\mathcal{K}_{12}$ (in Step 3). Let $U$ be a fixed guess of the elements in the intersection. In $\mathcal{P}^{(i)}_{j,U}$ we store (as before) the projection of the matchings from $\mathcal{L}^{(i)}_j$ on $U_1$ and $U_2$ — however, we are now only interested in matchings that satisfy $\mathcal{M}_{12} \cap \mathcal{K}_{12} = U$. In the next step, we remove the elements of $U$ from every set in $\mathcal{P}^{(i)}_{j,U}$ to obtain $\mathcal{D}^{(i)}_{j,U}$. Now, we compute an $\alpha(j,U)$-representative family of $\mathcal{D}^{(i)}_{j,U}$. Intuitively, since $\mathcal{M}_{12}$ already has $4(k-1)/3$ elements in common with $\mathcal{K}_{12}$, we can isolate these elements, set them aside, and compute representative families only for the remaining elements. Due to this restrictive computation of representative families, the amount of time we spend on this step improves considerably.

Towards correctness, we define $\mathcal{Q}^{(i)}$ as in Section 3. Moreover, it will be useful to define $\mathcal{Q}^{(i)}_{j,U}$, which further partitions the collection $\mathcal{Q}^{(i)}$ based on matching size and intersection with $\mathcal{K}_{12}$, i.e., $\mathcal{Q}^{(i)}_{j,U} := \{\mathcal{M} \mid \mathcal{M} \in \mathcal{Q}^{(i)}, |\mathcal{M}| = j, \mathcal{M}_{12} \cap \mathcal{K}_{12} = U\}$. Finally, let $\mathcal{X}^{(i)}_{j,U} := \{\mathcal{M}_{12} \mid \mathcal{M} \in \mathcal{Q}^{(i)}_{j,U}\}$. In this setting, our first claim states that if we have a Yes-instance of 3D-Matching Improvement, such that Lemma 5.1 is satisfied with respect to coordinates 1 and 2, then at least one relevant partial solution is preserved in $\mathcal{L}^{(i)}$ (at every stage of the algorithm).

Lemma 5.2. *For all $0 \leq i \leq n$, $U \subseteq \mathcal{K}_{12}$, and $0 \leq j \leq k$ such that $2(k+2)/3 - (2j - |U|) \geq 0$, we have that $\mathcal{R}^{(i)}_{j,U}$ is an $\alpha(j,U)$-representative family for $\mathcal{X}^{(i)}_{j,U}$, where $\alpha(j,U) = 2(k+2)/3 - (2j - |U|)$.*

*Proof.* The proof of this lemma, apart from Case 1, is the same as the proof of Lemma 3.1. Indeed, to see this, simply replace the notation in the proof of Lemma 3.1 with the notation in this section—that is, replace $2(k-j)$, $\mathcal{R}^{(i)}_j$, $\mathcal{X}^{(i)}_j$ and $\mathcal{P}^{(i)}_j$ by $\alpha(j,U)$, $\mathcal{R}^{(i)}_{j,U}$, $\mathcal{X}^{(i)}_{j,U}$ and $\mathcal{P}^{(i)}_{j,U}$, repectively. Next, we focus on the new proof of Case 1.

**Case 1.** $\mathcal{M}^Z$ contains a set with $c_{i+1}$ as the third coordinate.

Let $S \in \mathcal{M}^Z$ be such that $c_{i+1} \in S$. Define the smaller matching $\mathcal{M}^{Z \setminus S} := \mathcal{M}^Z \setminus S$. Notice that $|\mathcal{M}^{Z \setminus S}| = j - 1$ and $\lambda(\mathcal{M}^{Z \setminus S}) \leq i$. Let $U_S := U \cap S$ and $U' =$

**Algorithm 4:** Prune Partial Solutions $\mathcal{L}^{(i)}$

**for** $j \in \{0, 1, 2, \ldots, k\}$ **do**
  $\mathcal{L}_j^{(i)} \leftarrow \{\mathcal{M} \mid \mathcal{M} \in \mathcal{L}^{(i)}, |\mathcal{M}| = j\}$
  **for** $U \subseteq \mathcal{K}_{12}$ **do**
    $\alpha(j, U) \leftarrow \frac{2(k+2)}{3} - (2j - |U|)$
    **if** $\alpha(j, U) \geq 0$ **then**
      $\mathcal{P}_{j,U}^{(i)} \leftarrow \{\mathcal{M}_{12} \mid \mathcal{M} \in \mathcal{L}_j^{(i)} \text{ and } \mathcal{M}_{12} \cap \mathcal{K}_{12} = U\}$
      $\mathcal{D}_{j,U}^{(i)} \leftarrow \{X \setminus U \mid X \in \mathcal{P}_{j,U}^{(i)}\}$
      $\gamma \leftarrow \{(\mathcal{M}, \mathcal{M}_{12} \setminus U) \mid \mathcal{M} \text{ is a matching in } \mathcal{L}_j^{(i)} \text{ and } \mathcal{M}_{12} \setminus U \in \mathcal{D}_{j,U}^{(i)}\}$
      Compute $\mathcal{T}_{j,U}^{(i)} \subseteq_{rep}^{\alpha(j,U)} \mathcal{D}_{j,U}^{(i)}$ using Theorem 2.5
      $\mathcal{R}_{j,U}^{(i)} \leftarrow \{X \cup U \mid X \in \mathcal{T}_{j,U}^{(i)}\}$
  $\mathcal{L}_j^{(i)} \leftarrow \bigcup_{U \subseteq \mathcal{K}_{12}} \{\widetilde{\gamma}(\mathcal{M}_{12}) \mid \mathcal{M}_{12} \in \mathcal{R}_{j,U}^{(i)}\}$
$\mathcal{L}^{(i)} \leftarrow \bigcup_{j=1}^{k} \mathcal{L}_j^{(i)}$

$U \setminus U_S$. Therefore, $\mathcal{M}^{Z \setminus S} \in \mathcal{Q}_{j-1,U'}^{(i)}$ and $\mathcal{M}_{12}^{Z \setminus S} \in \mathcal{X}_{j-1,U'}^{(i)}$. Let $A = \mathcal{M}_{12}^{Z \setminus S}$ and $Y^S = Y \cup (\{S[1], S[2]\} \setminus U_S)$. It is easy to check that $A \cap Y^S = \emptyset$. We claim that $|Y^S| \leq \alpha(j-1, U')$:

$$|Y^S| = |Y| + 2 - |U_S| \leq \frac{2(k+1)}{3} - (2j - |U|) + 2 - |U_S| = \alpha(j-1, U')$$

Since $|U| = |U'| + |U_S| \leq |U'| + 2$ and $\alpha(j, U) \geq 0$, we have that $\alpha(j-1, U') \geq 0$. Hence, we are now in a situation where $A \in \mathcal{X}_{j-1,U'}^{(i)}$, $A \cap Y^S = \emptyset$, $|Y^S| \leq \alpha(j-1, U')$ and $\alpha(j-1, U') \geq 0$. By the induction hypothesis, we have that $\mathcal{R}_{j-1,U'}^{(i)}$ contains an $\alpha(j-1, U')$-representative of $A$ with respect to $Y^S$. Let us denote this representative by $B$. Note that $B \cap Y^S = \emptyset$ by definition. Also note that $B \cap U_S = \emptyset$ because $B \cap \mathcal{K}_{12} = U'$. Let $\mathcal{M}^B$ be the matching associated with $B$. Since $B \in \mathcal{R}_{j-1,U'}^{(i)}$, we have that $\mathcal{M}^B \in \mathcal{L}_{j-1}^{(i)}$. Since the $1,2$-coordinates of $S$ are disjoint from $B$ and the third coordinate of $S$ is $c_{i+1}$, we have that $\mathcal{M}^B \cap S = \emptyset$. Thus, the matching $\mathcal{M}^B \cup S \in \mathcal{L}_j^{(i+1)}$. Let us denote this matching by $\mathcal{M}^{B|S}$. It is easy to check that $\mathcal{M}_{12}^{B|S} \cap \mathcal{K}_{12} = U$, and correspondingly, $\mathcal{M}_{12}^{B|S} \in \mathcal{P}_{j,U}^{(i+1)}$. Let $V := \mathcal{M}_{12}^{B|S} \setminus U$. Note that $V \in \mathcal{D}_{j,U}^{(i+1)}$ and $V \cap Y = \emptyset$. Since $\mathcal{T}_{j,U}^{(i+1)}$ is an $\alpha(j, U)$-representative family for $\mathcal{P}_{j,U}^{(i+1)}$, we have that $\mathcal{T}_{j,U}^{(i+1)}$ contains an $\alpha(j, U)$-representative $V^\star$ for $V$ with respect to $Y$. We now define $Z^\star$ to be $V^\star \cup U$. Note that $Z^\star \in \mathcal{R}_{j,U}^{(i+1)}$. Since $V^\star$ is disjoint from $Y$ by definition, and we know that $U \cap Y = \emptyset$, we conclude that $Z^\star$ is the desired representative.
□

We now establish the correctness of 3-DM Algorithm; the proof is a straightforward modification of the proof of Lemma 3.2.

Lemma 5.3. *Let $(U_1, U_2, U_3, \mathcal{F}, \mathcal{K})$ be a* Yes*-instance of* 3D-Matching Improvement. *Then,* 3-DM Algorithm *successfully computes a 3D-Matching of size $k$.*

Finally, we turn to an analysis of the running time of Algorithm 3-DM Algorithm.

Lemma 5.4. 3-DM Algorithm *runs in time* $\mathcal{O}(2.0035^{3k} \cdot |\mathcal{F}| \cdot n \log^2 n)$.

*Proof.* Let $1 \leq i \leq n$ and $1 \leq j \leq k$ be fixed, and let us consider the running time of Step 9 of the Algorithm 4, which is the computation of an $\alpha(j, U)$-representative family, where $U \subseteq \mathcal{K}_{12}$. By Theorem 2.5, the running time of this computation for any $U \subseteq \mathcal{K}_{12}$ is bounded by:

$$\left(\frac{2(k+2)/3}{2(k+2)/3 - (2j - |U|)}\right)^{2(k+2)/3-(2j-|U|)} \cdot 2^{o(k)} \cdot |\mathcal{D}^{(i)}_{j,U}| \cdot \log n$$

In the $i^{th}$ iteration, observe that for any $U \subseteq \mathcal{K}_{12}$, the size of the family $\mathcal{D}^{(i)}_{j,U}$ is same as the size of the family $\mathcal{P}^{(i)}_{j,U}$. Note that the size of the family $\mathcal{P}^{(i)}_{j,U}$ is given by the size of the representative families computed in the previous iteration: first, those that involved matchings of size $j$ whose intersection with $\mathcal{K}_{12}$ is exactly $U$, and second, those that involved matchings of size $(j-1)$ and were updated to matchings of size $j$ whose intersection with $\mathcal{K}_{12}$ will be exactly $U$. For a set $S \in \mathcal{F}$ and a disjoint matching $\mathcal{M}$, let $\mathcal{M}^S$ be the matching $\mathcal{M} \cup S$. Now, it is easy to see that for any $S \in \mathcal{F}$ and $\mathcal{M} \in \mathcal{L}^{(i-1)}$, $\mathcal{M}^S_{12}$ is added to $\mathcal{P}^{(i)}_{j,U}$ only if $\mathcal{M}_{12} \cap \mathcal{K}_{12} = U \setminus (S \cap \mathcal{K}_{12})$ and $|\mathcal{M}| = j - 1$. In other words, for a fixed $S$, if $\mathcal{M}^S_{12} \in \mathcal{P}^{(i)}_{j,U}$, then $\mathcal{M}_{12} \in \mathcal{R}^{(i-1)}_{j-1,U'}$, where $U' = U \setminus (S \cap \mathcal{K}_{12})$. Therefore,

$$\begin{aligned}|\mathcal{D}^{(i)}_{j,U}| &= |\mathcal{P}^{(i)}_{j,U}| \\ &\leq |\mathcal{R}^{(i-1)}_{j,U}| + |\mathcal{F}| \cdot \left(\max_{U' \subseteq U} \left\{|\mathcal{R}^{(i-1)}_{j-1,U'}| \ : \ |U \setminus U'| \leq 2\right\}\right)\end{aligned}$$

Again, by Theorem 2.5, we can upper bound the sizes of the representative families, and thus for any $U$:

$$|\mathcal{D}^{(i)}_{j,U}| = \mathcal{O}\left(\binom{2(k+2)/3}{2j - |U|} 2^{o(k)} \cdot |\mathcal{F}| \cdot \log n\right)$$

For fixed $i$ and $j$, we compute a representative family for each subset $U \in \mathcal{K}_{12}$, and thus runs $2^{2k}$ times. Thus, the running time for computation of all the representative sets for some fixed $i$ is bounded by:

$$4^k \max_{\substack{1 \leq j \leq k \\ U \subseteq \mathcal{K}_{12}}} \mathcal{O}\left(\binom{2(k+2)/3}{2j - |U|} \cdot \left(\frac{2(k+3)/3}{2(k+2)/3 - (2j - |U|)}\right)^{2(k+2)/3-(2j-|U|)}\right) \cdot 2^{o(k)} |\mathcal{F}| \cdot \log^2 n$$

The maximum is achieved at $2j - |U| = \beta \cdot 2(k+2)/3$, where $\beta = 1 + \dfrac{1 - \sqrt{1+4e}}{2e}$. Therefore, the running time above simplifies to $\mathcal{O}(4^k \cdot (2.8505)^{2k/3} \cdot |\mathcal{F}| \cdot n \log^2 n)$. Note that the algorithm runs three times: once for each choice of the pair of $p$ and $q$. Moreover, Algorithm 4 runs at most $n$ times. Thus, the overall running time is $\mathcal{O}(4^k \cdot (2.8505)^{2k/3} \cdot |\mathcal{F}| \cdot n \log^2 n) = \mathcal{O}(2.0035^{3k} \cdot |\mathcal{F}| \cdot n \log^2 n)$. □

Putting together Lemmas 5.2, 5.3, and 5.4, and using the fact that 3D-Matching can be solved by invoking 3D-Matching Improvement at most $k$ times, we have the following theorem.

Theorem 5.5. $(3,k)$-DM *can be solved in time* $\mathcal{O}(2.0035^{3k} \cdot |\mathcal{F}| \cdot n \log^2 n)$.

**6. An Algorithm for $r$-Set $k$-Packing.** In this section, we describe an algorithm with running time $\mathcal{O}(2.851^{(r-0.5501)k} \cdot |\mathcal{F}| \cdot n \log^2 n \cdot \log W)$ for the $(r,k)$-WSP problem. Recall that in the algorithm for $(r,k)$-WDM, we achieved an improved running time by handling the elements in $U_r$ differently than the rest of the elements in $U$. We iterated over the elements $c_1, c_2, \ldots, c_n$ in $U_r$, where in the $i^{th}$ iteration, we only stored matchings whose maximum last index is at most $i$. Since each set in $\mathcal{F}$ contains exactly one element from $U_r$, we concluded that: (1) the sets to be added to any partial matching computed and stored at the $i^{th}$ iteration, do not contain elements from $\{c_1, c_2, \ldots, c_i\}$, and (2) any partial solution (partial matching) does not contain elements from $\{c_{i+1}, c_{i+2}, \ldots, c_n\}$. Now, in the $(r,k)$-WSP problem, we do not have such a set $U_r$ which we can handle in the same manner. However, we can still use a similar approach. Again, we will iterate over all elements $c_1, c_2, \ldots, c_n$ in the universe $U$, such that in the $i^{th}$ iteration, we will be able to conclude that the sets to be added to partial solutions (partial set packings) do not contain elements from $\{c_1, c_2, \ldots, c_i\}$. We will not be able to make the above mentioned second conclusion, and thus the running time of our algorithm for $(r,k)$-WSP is not as good as the running time of the algorithm for $(r,k)$-WDM; yet, it clearly improves upon the standard application of representative families.

Let $(U, \mathcal{F}, w : \mathcal{F} \to \mathbb{N}, k, W)$ be the given instance of $(r,k)$-WSP, and let $U = \{c_1, c_2, \ldots, c_n\}$. Given a set $S \in \mathcal{F}$, we define $S[i]$ as the $i^{th}$ largest index element in $S$. In particular, $S[r] = c_j$ if $S \cap \{c_1, \ldots, c_j\} = \{c_j\}$ (i.e, $c_j$ is the smallest index element in $S$). Let $S_{[r-1]} = \{S[i] \mid 1 \leq i \leq r-1\}$, $\mathcal{M}_{[r-1]} := \bigcup_{S \in \mathcal{M}} S_{[r-1]}$ and $\mathcal{M}_r := \{S[r] \mid S \in \mathcal{M}\}$, and define the maximum last index of $\mathcal{M}$ as $\lambda(\mathcal{M}) := \max\{i \mid c_i \in \mathcal{M}\}$. Correspondingly, $\gamma_j := \{(\mathcal{M}, \mathcal{M}_{[r-1]}) \mid \mathcal{M} \text{ is a packing of size } j \text{ in } \mathcal{F}\}$. Now, we define $\mathcal{Q}^{(i)}$, $\mathcal{Q}^{(i)}_j$ and $\widetilde{\gamma}_j(S)$ exactly as before. The following simple observation implies that we can indeed apply the above mentioned approach.

Observation 1 ([5]). *Let $\mathcal{M}$ be a packing. Then, any set $S \in \mathcal{F}$ satisfying $S[r] > \lambda(\mathcal{M})$, also satisfies $S \cap \mathcal{M}_r = \emptyset$.*

By using the above notation, our algorithm for $(r,k)$-WSP, Algorithm 5, is *identical* to the algorithm for $(r,k)$-WDM, apart from the parameter used to compute representative family—we compute max $r(k-j)$-representative families instead of $(r-1)(k-j)$-representative families. This is due to the following reason. In $\mathcal{L}^{(i)}$, we store carefully chosen packings whose maximum last index is at most $i$, and we cannot assume that the packing in $\mathcal{L}^{(i)}$ do not contain the elements $c_{i+1}, c_{i+2}, \ldots, c_n$. In particular, this implies that by computing an $(r-1)(k-j)$-representative family, we cannot assume that there is a packing in $\mathcal{L}^{(i)}_j$ that can be completed to a solution (if a solution exists), but by computing an $r(k-j)$-representative family, we can assume this, since $r(k-j)$ is the size of any set using which a packing in $\mathcal{L}^{(i)}_j$ might be completed to a solution. Therefore, the correctness of Algorithm 5 follows in the same manner as the correctness of our algorithm for $(r,k)$-WDM.

Finally, we state the running time of Algorithm 5.

Lemma 6.1. *The running time of Algorithm 5 is bounded by* $\mathcal{O}(2.85043^{(r-0.5501)k} \cdot$

$|\mathcal{F}| \cdot n \log^2 n \cdot \log W)$. *When* $r = 3$*, Algorithm 5 runs in time* $\mathcal{O}(12.15493^k \cdot |\mathcal{F}| \cdot n \log^2 n \cdot \log W)$.

*Proof.* Let $1 \leq i \leq n$ and $1 \leq j \leq (k-1)$ be fixed. Let $j' = j - 1$. First, consider the running time of Step 5 in the module **Compute Partial Solutions** $\mathcal{L}^{(i)}$ (in the context of Algorithm 5). Due to Corollary 2.7, it is bounded by

$$2^{o(k)} \log^2 n \max_{1 \leq j'' < r} \left\{ \binom{rk-j}{(r-1)j'+j''} \cdot \left( \frac{rk-j}{(r-1)j'+j''} \right)^{(rk-(rj+r-1-j''))} + \binom{rk-j}{(r-1)j'+j''} \cdot \log W \right\}$$

By Corollary 2.7 and Theorem 2.6, $|\mathcal{P}_j^{(i)}| \leq (|\mathcal{F}|+1) \cdot \binom{rk-j}{(r-1)j} 2^{o(rk)} \log n$. Thus, again by Theorem 2.6, the running time of Step 8 of the module **Prune Partial Solutions** $\mathcal{L}^{(i)}$ (in the context Algorithm 5 is) bounded by:

$$\mathcal{O}\left( \left( \frac{rk-j}{r(k-j)} \right)^{r(k-j)} \cdot 2^{o(rk)} \cdot \binom{rk-j}{(r-1)j} \cdot |\mathcal{F}| \cdot \log^2 n \cdot \log W \right).$$

Denote $x = 2^{o(rk)} \cdot |\mathcal{F}| \cdot n \log^2 n \cdot \log W$. Thus, the overall running time can be bounded by:

$$\mathcal{O}(x \cdot \max_{0<\alpha<1} \left\{ \frac{(rk-\alpha k)^{2rk-\alpha rk-\alpha k}}{(\alpha rk-\alpha k)^{\alpha rk-\alpha k}(rk-\alpha rk)^{2rk-2\alpha rk}} \right\})$$

$$= \mathcal{O}(x \cdot \max_{0<\alpha<1} \left\{ \left[ \frac{(r-\alpha)^{2r-\alpha r-\alpha}}{(\alpha r-\alpha)^{\alpha r-\alpha}(r-\alpha r)^{2r-2\alpha r}} \right]^k \right\})$$

$$= \mathcal{O}(x \cdot \left[ \max_{0<\alpha<1} \left\{ \left( \frac{\alpha r-\alpha}{r-\alpha} \right)^{\alpha} \left( \frac{(r-\alpha)^{2-\alpha}}{(\alpha r-\alpha)^{\alpha}(r-\alpha r)^{2-2\alpha}} \right)^{r} \right\} \right]^k) = (*)$$

When $q = 3$, the maximum of (*) is achieved at $\alpha \cong 0.58226$. Thus, Algorithm 5

solves $(3,k)$-WSP in time $O^*(12.15493^k)$. Now, we can simplify (*) as follows

$$
\begin{aligned}
(*) &= \mathcal{O}\left(x \max_{0<\alpha<1}\left[\left(\frac{\alpha r-\alpha}{r-\alpha}\right)^{\alpha}\left(\frac{(r-\alpha)^{2-\alpha}}{(\alpha r-\alpha)^{\alpha}(r-\alpha r)^{2-2\alpha}}\right)^{r}\right]^{k}\right)\\
&= \mathcal{O}\left(x \max_{0<\alpha<1}\left[\alpha^{\alpha}\left(\frac{1}{\alpha^{\alpha}(1-\alpha)^{2-2\alpha}}\right)^{r}\cdot\left(\frac{r-1}{r-\alpha}\right)^{\alpha}\left(\frac{(r-\alpha)^{2-\alpha}}{(r-1)^{\alpha}r^{2-2\alpha}}\right)^{r}\right]^{k}\right)\\
&= \mathcal{O}\left(x \max_{0<\alpha<1}\left[\alpha^{\alpha}\left(\frac{1}{\alpha^{\alpha}(1-\alpha)^{2-2\alpha}}\right)^{r}\cdot\left(\frac{r-\alpha}{r-1}\right)^{-\alpha}\left(\frac{r-\alpha}{r-1}\right)^{\alpha r}\left(\frac{r-\alpha}{r}\right)^{(2-2\alpha)r}\right]^{k}\right)\\
&= \mathcal{O}\left(x \max_{0<\alpha<1}\left[\alpha^{\alpha}\left(\frac{1}{\alpha^{\alpha}(1-\alpha)^{2-2\alpha}}\right)^{r}\cdot\left(1+\frac{1-\alpha}{r-1}\right)^{\alpha(r-1)}\left(1-\frac{\alpha}{r}\right)^{(2-2\alpha)r}\right]^{k}\right)\\
&= \mathcal{O}\left(x \max_{0<\alpha<1}\left[\alpha^{\alpha}\left(\frac{1}{\alpha^{\alpha}(1-\alpha)^{2-2\alpha}}\right)^{r}\cdot e^{\alpha(1-\alpha)}\cdot e^{-\alpha(2-2\alpha)}\right]^{k}\right)\\
&= \mathcal{O}\left(x \max_{0<\alpha<1}\left[\left(\frac{\alpha}{e^{1-\alpha}}\right)^{\alpha}\left(\frac{1}{\alpha^{\alpha}(1-\alpha)^{2-2\alpha}}\right)^{r}\right]^{k}\right)
\end{aligned}
$$

As we increase $r$, the $\alpha$ for which we get the maximum decreases, staying greater than $\alpha^* = 1+\frac{1-\sqrt{1+4e}}{2e}$ (since this $\alpha^*$ maximizes $(\frac{1}{\alpha^{\alpha}(1-\alpha)^{2-2\alpha}})^r$). For example when $r = 1,500$, the maximum of (*) is achieved at $\alpha' < 0.550148$, and thus when $r \geq 1,500$, we get that Algorithm 5 runs in time,

$$
\begin{aligned}
\mathcal{O}(x\cdot(\frac{\alpha'}{e^{1-\alpha'}})^{\alpha'}(\frac{1}{\alpha^{*\alpha^*}(1-\alpha^*)^{2-2\alpha^*}})^r) &= \mathcal{O}(x\cdot(0.56201\cdot 2.85043^r)^k)\\
&= \mathcal{O}(x\cdot 2.85043^{(r-0.5501)k})
\end{aligned}
$$

Since this expression bounds (*) for smaller values for $r$, we get the desired running time. □

We have thus obtained the following theorem.

THEOREM 6.2. *The* $(r,k)$-WSP *problem can be solved in time* $\mathcal{O}(2.851^{(r-0.5501)k}\cdot|\mathcal{F}|\cdot n\log^2 n\cdot\log W)$. *Moreover, the special case of* $(3,k)$-WSP *can be solved in time* $\mathcal{O}(12.155^k\cdot|\mathcal{F}|\cdot n\log^2 n\cdot\log W)$.

**7. Kernels.** In this section we present a kernelization algorithm for $(r,k)$-WSP. Since $(r,k)$-WDM is a special case of $(r,k)$-WSP, the same algorithm works for $(r,k)$-WDM as well. Thus, we prove the following result.

THEOREM 7.1. *The* $(r,k)$-WSP *and* $(r,k)$-WDM *problems admit a kernel of size* $\mathcal{O}(e^r(k-1)^r\log W)$.

*Proof.* Let $(U,\mathcal{F},w:\mathcal{F}\to\mathbb{N},k,W)$ be an instance of $(r,k)$-WSP. We assume without loss of generality that $U$ does not contain elements that do not appear in any set in $\mathcal{F}$, and that the maximum weight in the image of $w$ is at most $W$. The kernelization algorithm works as follows. If $|\mathcal{F}^*| \leq \binom{rk}{r}$, then the algorithm returns the given input instance as the kernel. Otherwise, the algorithm computes a max $(rk-r)$-representative family $\mathcal{F}^* \subseteq^{rk-r}_{maxrep} \mathcal{F}$ using Theorem 2.4, and outputs an instance

$(U^*, \mathcal{F}^*, w^* : \mathcal{F}^* \to \mathbb{N}, k, W)$ where $U^* \leftarrow \bigcup_{S \in \mathcal{F}^*} S$ and $w^*$ is the function $w$ restricted to the domain $\mathcal{F}^*$. Since $|\mathcal{F}^*| \leq \binom{rk}{r}$ and $U^* \leftarrow \bigcup_{S \in \mathcal{F}^*} S$, the size of output instance is bounded by $\mathcal{O}(\binom{rk}{r}) = \mathcal{O}(e^r(k-1)^r \log W)$ (the factor $\log W$ is added since the output includes the image of $w$). The running time of the algorithm is bounded by a polynomial in $\mathcal{F}$ and $\log W$.

Now, we prove the correctness of the algorithm. If $(U, \mathcal{F}, w : \mathcal{F} \to \mathbb{N}, k, W)$ is a NO-instance, then clearly $(U^*, \mathcal{F}^*, w^* : \mathcal{F}^* \to \mathbb{N}, k, W)$ is a NO-instance because $\mathcal{F}^* \subseteq \mathcal{F}$. Next, suppose that $(U, \mathcal{F}, w : \mathcal{F} \to \mathbb{N}, k, W)$ is a YES-instance. We need to prove that $(U^*, \mathcal{F}^*, w^* : \mathcal{F}^* \to \mathbb{N}, k, W)$ is a YES-instance. Let $P$ be a packing in $\mathcal{F}$ of size $k$, of weight at least $W$, such that $|P \cap \mathcal{F}^*|$ is maximized. We claim that $P \subseteq \mathcal{F}^*$. Suppose not, then there exists a set $S \in (P \setminus \mathcal{F}^*)$. By Theorem 2.4, there exists a set $S^* \in \mathcal{F}^*$ such that $S^* \cap (\bigcup_{S' \in (P \setminus \{S\})} S') = \emptyset$ and $w(S^*) \geq w(S)$. Thus, $P' = (P \setminus \{S\}) \cup \{S^*\}$ is a set packing of size $k$ in $\mathcal{F}$ of weight at least $W$. This is a contradiction to our assumption that $|P \cap \mathcal{F}^*|$ is maximized. □

**8. Conclusions.** We have demonstrated that by identifying and carefully exploiting structural properties of problems of interest, representative families can be used to a great advantage inside dynamic programming routines. Towards obtaining our results, we relied also on the iterative expansion technique, as well as a compact computation of representative sets (see Corollary 2.7). Thus, we were able to significantly improve on the state of the art for the $r$-DIMENSIONAL $k$-MATCHING and $r$-SET $k$-PACKING problems (even with weights), along with a further improved running time for the special case of the 3D-MATCHING problem.

It will be very interesting to see whether other seemingly unrelated classical techniques, such as refined memorization [4] and the method of bounded search trees (see [12]), can be combined with computations of representative sets.

**Acknowledgements.** We would like to thank Saket Saurabh for directing us to this problem, and providing several useful pointers.

REFERENCES

[1] N. Alon, R. Yuster, and U. Zwick. Color-coding. *J. ACM*, 42(4):844–856, July 1995.
[2] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. Narrow sieves for parameterized paths and packings. *CoRR*, abs/1007.1161, 2010.
[3] B. Bollobás. On generalized graphs. *Acta Math. Acad. Sci. Hungar*, 16:447–452, 1965.
[4] L. S. Chandran and F. Grandoni. Refined memorization for vertex cover. *Inf. Process. Lett.*, 93(3):123–131, 2005.
[5] J. Chen, Q. Feng, Y. Liu, S. Lu, and J. Wang. Improved deterministic algorithms for weighted matching and packing problems. *Theor. Comput. Sci.*, 412(23):2503–2512, 2011.
[6] J. Chen, D. Friesen, W. Jia, and I. Kanj. Using nondeterminism to design effcient deterministic algorithms. *Algorithmica*, 40(2):83–97, 2004.
[7] J. Chen, J. Kneis, S. Lu, D. Molle, S. Richter, P. Rossmanith, S. H. Sze, and F. Zhang. Randomized divide-and-conquer: Improved path, matching, and packing algorithms. *SIAM J. on Computing*, 38(6):2526–2547, 2009.
[8] J. Chen, Y. Liu, S. Lu, S.-H. Sze, and F. Zhang. Iterative expansion and color coding: An improved algorithm for 3D-matching. *ACM Transactions on Algorithms*, 8(1):6, 2012.
[9] S. Chen and Z. Chen. Faster deterministic algorithms for packing, matching and t-dominating set problems. *CoRR*, abs/1306.3602, 2013.
[10] H. Dell and D. Marx. Kernelization of packing problems. In *SODA*, pages 68–81, 2012.
[11] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
[12] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.

[13] R. G. Downey, M. R. Fellows, and M. Koblitz. Techniques for exponential parameterized reductions in vertex set problems. (Unpublished, reported in [11], §8.3).
[14] M. R. Fellows, C. Knauer, N. Nishimura, P. Ragde, F. A. Rosamond, U. Stege, D. M. Thilikos, and S. Whitesides. Faster fixed-parameter tractable algorithms for matching and packing problems. *Algorithmica*, 52(2):167–176, 2008.
[15] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Text in Theoretical Computer Science. Springer, 2006.
[16] F. V. Fomin, D. Lokshtanov, F. Panolan, and S. Saurabh. Representative sets of product families. In *ESA*, pages 443–454, 2014.
[17] F. V. Fomin, D. Lokshtanov, and S. Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. In *SODA*, pages 142–151, 2014.
[18] M. R. Garey and D. S. Johnson. *Computers and Intractability*. Freeman, San Francisco, 1979.
[19] P. Goyal, N. Misra, and F. Panolan. Faster deterministic algorithms for r-dimensional matching using representative sets. In *FSTTCS*, pages 237–248, 2013.
[20] D. Hermelin and X. Wu. Weak compositions and their applications to polynomial lower bounds for kernelization. In *SODA*, pages 104–113, 2012.
[21] I. Koutis. A faster parameterized algorithm for set packing. *Information Processing Letters*, 94:7–9, 2005.
[22] I. Koutis. Faster algebraic algorithms for path and packing problems. In *Proceedings of Automata, Languages and Programming, 35th International Colloquium, ICALP*, volume 5125 of *Lecture Notes in Computer Science*, pages 575–586. Springer, 2008.
[23] I. Koutis and R. Williams. Limits and applications of group algebras for parameterized problems. In *Proceedings of Automata, Languages and Programming, 36th International Colloquium, ICALP*, volume 5555 of *LNCS*, pages 653–664. Springer, 2009.
[24] Y. Liu, J. Chen, and J. Wang. On efficient FPT algorithms for weighted matching and packing problems. In *TAMC*, pages 575–586, 2007.
[25] Y. Liu, S. Lu, J. Chen, and S.-H. Sze. Greedy localization and color-coding: improved matching and packing algorithms. In *International Workshop on Parameterized and Exact Computation IWPEC*, volume 4169 of *LNCS*, pages 84–95. Springer, 2006.
[26] D. Marx. Parameterized coloring problems on chordal graphs. *Theor. Comput. Sci.*, 351:407–424, 2006.
[27] B. Monien. How to find long paths efficiently. *Ann. Discrete Math.*, 25:239–254, 1985.
[28] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.
[29] R. Y. Pinter, H. Shachnai, and M. Zehavi. Kernelization of packing problems. In *MFCS*, pages 589–600, 2014.
[30] H. Shachnai and M. Zehavi. Representative families: a unified tradeoff-based approach. In *ESA*, pages 786–797, 2014.
[31] J. Wang and Q. Feng. Improved parameterized algorithms for weighted 3-set packing. In *Proc. COCOON*, pages 130–139, 2008.
[32] J. Wang and Q. Feng. An $O^*(3.523^k)$ parameterized algorithm for 3-set packing. In *TAMC*, pages 82–93, 2008.
[33] V. V. Williams. Multiplying matrices faster than Coppersmith – Winograd. In *SODA*, pages 887–898, 2012.