Contents lists available at ScienceDirect

Theoretical Computer Science

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

Improved deterministic algorithms for weighted matching and packing problems[☆]

Jianer Chen[a,b], Qilong Feng[a], Yang Liu[c], Songjian Lu[b], Jianxin Wang[a,*]

[a] *School of Information Science and Engineering, Central South University, Changsha 410083, PR China*
[b] *Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843, USA*
[c] *Department of Computer Science, University of Texas-Pan American, Edinburg, TX 78539, USA*

A R T I C L E I N F O

Keywords:
rD-MATCHING
r-SET PACKING
Randomized divide-and-conquer
(n, k)-universal set

A B S T R A C T

Based on the method of (n, k)-universal sets, we present a deterministic parameterized algorithm for the weighted rD-MATCHING problem with time complexity $O^*(4^{(r-1)k+o(k)})$, improving the previous best upper bound $O^*(4^{rk+o(k)})$. In particular, the algorithm applied to the unweighted 3D-MATCHING problem results in a deterministic algorithm with time $O^*(16^{k+o(k)})$, improving the previous best result $O^*(21.26^k)$. For the weighted r-SET PACKING problem, we present a deterministic parameterized algorithm with time complexity $O^*(2^{(2r-1)k+o(k)})$, improving the previous best result $O^*(2^{2rk+o(k)})$. The algorithm, when applied to the unweighted 3-SET PACKING problem, has running time $O^*(32^{k+o(k)})$, improving the previous best result $O^*(43.62^{k+o(k)})$. Moreover, for the weighted r-SET PACKING and weighted rD-MATCHING problems, we give a kernel of size $O(k^r)$, which is the first kernelization algorithm for the problems on weighted versions.

1. Introduction

Matching and packing problems form an important class of NP-hard problems. In particular, the 3D-MATCHING problem is one of the six "basic" NP-complete problems [7]. In this paper, we are mainly focused on the weighted rD-MATCHING and weighted r-SET PACKING problems, for $r \geq 3$, which are formally defined as follows.

Weighted rD-MATCHING: Given a collection $S \subseteq A_1 \times A_2 \times \cdots \times A_r$ of r-tuples and an integer k, where $A_1, A_2, \ldots, A_r$ are pair-wise disjoint sets and each r-tuple in S has a weight value, find a subcollection S' of k r-tuples in S with the maximum weight sum such that no two r-tuples in S' have common elements, or report that no such subcollection exists.

Weighted r-SET PACKING: Given a collection S of r-sets (i.e., sets that contain exactly r elements) and an integer k, where each r-set in S has a weight value, find a subcollection S' of k r-sets in S with the maximum weight sum such that no two r-sets in S' have common elements, or report that no such subcollection exists.

Parameterized algorithms for rD-MATCHING and r-SET PACKING have been an active research line, whose running time are bounded by a polynomial of the input size n times a function $f(k)$ of the parameter k. First note that the r-SET PACKING

☆ A preliminary version of this work was reported in the Proceedings of the 6th Annual Conference on Theory and Applications of Models of Computation, Lecture Notes in Computer Science, vol. 5532, 2009, pp. 211–220. This work is supported in part by the National Science Foundation of USA under grant CCF-0830455, by the National Natural Science Foundation of China (No. 60773111, 61073036, 61070224), the Doctoral Discipline Foundation of Higher Education Institution of China under Grant 20090162110056.

* Corresponding author.
E-mail address: jxwang@mail.csu.edu.cn (J. Wang).

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.10.042

Table 1
Comparison of deterministic algorithms for matching.

References	Time complexity	3DM	rDM	W3DM	WrDM	Remark
[4]	$O^*((rk)!(rk)^{3rk})$	✓	✓	✓	✓	
[5]	$O^*(2^{5rk-4k}\binom{6(r-1)k+k}{rk})$	✓	✓	✓	✓	
[11]	$O^*(12.8^{rk})$	✓	✓	✓	✓	
[2]	$O^*(4^{rk+o(k)})$	✓	✓	✓	✓	
Our result	$O^*(4^{(r-1)k+o(k)})$	✓	✓	✓	✓	
[15]	$O^*(432^{k+o(k)})$	✓		✓		$r = 3$ only
Our result	$O^*(16^{k+o(k)})$	✓		✓		$r = 3$
[1]	$O^*((r-1)^k((r-1)k/e)^{k(r-2)})$	✓	✓	×	×	
[9]	$O^*(2^{O(rk)})$	✓	✓	×	×	
[16]	$O^*(43.62^{k+o(k)})$	✓		×		$r = 3$ only
[12]	$O^*(21.26^k)$	✓		×		$r = 3$ only
[10]	$O^*(8^k)$ (randomized)	✓		×		$r = 3$ only

problem is a general extension of the rD-MATCHING problem. Therefore, all parameterized algorithms for r-SET PACKING can also be used to solve rD-MATCHING. The weighted rD-MATCHING and weighted r-SET PACKING problems were first studied by Downey and Fellows [4], where a deterministic parameterized algorithm of time complexity $O^*((rk)!(rk)^{3rk})$ was presented.[1] An improved deterministic algorithm for the weighted rD-MATCHING and weighted r-SET PACKING problems was proposed by Fellows et al. [5] with running time $O^*(2^{5rk-4k}\binom{6(r-1)k+k}{rk})$. Liu et al. [11] further reduced the time complexity to $O^*(12.8^{rk})$. For weighted 3D-MATCHING, Wang and Feng [15] gave a more efficient algorithm of time complexity $O^*(7.56^{3k+o(k)}) = O^*(432^{k+o(k)})$. Currently, the best deterministic algorithms for the weighted rD-MATCHING and weighted r-SET PACKING problems are due to Chen et al. [2] with time complexity $O^*(4^{rk+o(k)})$.

We remark that there is also a very active research line on parameterized algorithms for unweighted versions of the problems. Using the technique of Greedy Localization, Chen et al. [1] and Jia et al. [8] presented deterministic algorithms for the unweighted rD-MATCHING and r-SET PACKING problems of time $O^*((r-1)^k((r-1)k/e)^{k(r-2)})$. Koutis [9] gave an improved deterministic algorithm of time $O^*(2^{O(rk)})$. For the case of $r = 3$, Liu et al. [12] presented a deterministic algorithm of time $O^*(21.26^k)$ for the unweighted 3D-MATCHING problem, which is currently the best deterministic algorithm for the unweighted 3D-MATCHING problem. For the unweighted 3-SET PACKING problem, Liu et al. [12] gave a deterministic algorithm of time $O^*(97.98^k)$, which was further improved by Wang and Feng [16] with an algorithm of time $O^*(43.62^{k+o(k)})$. This is currently the best deterministic algorithm for the unweighted 3-SET PACKING problem. It should be pointed out that all these algorithms [1,8,9,12,16] seem to work only for unweighted versions and cannot be applied to solve the weighted versions. Very recently, Koutis [10] proposed a randomized algorithm of time complexity $O^*(8^k)$ for the unweighted 3D-MATCHING and 3-SET PACKING problems. However, as remarked in [17], the algorithms in [10] only work for unweighted case and do not appear to extend to the weighted versions. Moreover, whether the algorithms can be extended to rD-MATCHING and r-SET PACKING for $r > 3$, and whether the randomized algorithms can be derandomized are still unknown.

In this paper, we study the weighted versions of rD-MATCHING and r-SET PACKING, and develop a different algorithmic approach to the problems. In particular, after further analyzing the structure of the problems and using (n, k)-universal sets and the divide-and-conquer methods, we are able to develop a deterministic algorithm of time $O^*(4^{(r-1)k+o(k)})$ for the weighted rD-MATCHING problem, improving the previous best result $O^*(4^{rk+o(k)})$. In fact, our deterministic algorithms for the weighted cases are even better than the previous best deterministic algorithm for the unweighted cases. For example, our algorithm for weighted 3D-MATCHING runs in time $O^*(16^{k+o(k)})$, which is much better than the previous best algorithm of time $O^*(21.26^k)$ for unweighted 3D-MATCHING.

Table 1 provides a comprehensive comparison of deterministic algorithms for the rD-MATCHING problems, where "✓" denotes that the algorithm can be applied to the problem, "×" denotes that the algorithm is not applicable to the problem, and 3DM, W3DM, rDM, and WrDM denote unweighted 3D-MATCHING, weighted 3D-MATCHING, unweighted rD-MATCHING, and weighted rD-MATCHING, respectively. We have partitioned the algorithms into three groups. In the first group (the first 5 rows in the table), we compare our algorithm with previous algorithms that are applicable to rD-MATCHING for a general integer $r \geq 3$ and for both weighted and unweighted versions. In the second group, consisting of rows 6–7 in the table, we compare our algorithm with the best previous algorithm for 3D-MATCHING for both weighted and unweighted versions. Finally, the last group, consisting of rows 8–12 in the table, lists all previous algorithms that are only applicable to unweighted version of rD-MATCHING. As we can see, our algorithm improves all previous algorithms significantly, except the last row, which, as we have mentioned above, seems difficult to be derandomized and/or extended to general integer $r > 3$.

For the weighted r-SET PACKING problem, we also develop an improved deterministic algorithm. By further analyzing the structure of the problem, we present an algorithm of time $O^*(2^{(2r-1)k+o(k)})$ for weighted r-SET PACKING, improving the

[1] Following the recent convention, for a given function f, we will use the notion $O^*(f)$ for $O(f \cdot n^{O(1)})$.

previous best result $O^*(2^{2rk+o(k)})$. In particular, for the case of $r = 3$, our deterministic algorithm for weighted 3-SET PACKING runs in time $O^*(32^{k+o(k)})$, which is even better than the previous best algorithm of time $O^*(43.62^{k+o(k)})$ for the unweighted 3-SET PACKING problem.

Finally, for the weighted r-SET PACKING and rD-MATCHING problems, we develop a polynomial time kernelization algorithm that gives the problems a kernel of size $O(k^r)$. This gives the first kernelization algorithm for the problems for their weighted versions.

Before we start our discussion, we give a brief review on the necessary background.

Assume that n and k are integers such that $n > k$. Let Z_n be the set $\{0, 1, \ldots, n-1\}$. A *splitting function* over Z_n is a $\{0, 1\}$ function over Z_n. A subset W of Z_n is called a *k-subset* if it contains exactly k elements. Let (W_0, W_1) be a partition of the k-subset W, i.e., $W_0 \cup W_1 = W$ and $W_0 \cap W_1 = \emptyset$. We say that a splitting function f over Z_n *implements* the partition (W_0, W_1) if $f(x) = 0$ for all $x \in W_0$ and $f(y) = 1$ for all $y \in W_1$.

Definition (*[13]*). A set $\Psi_{n,k}$ of splitting functions over Z_n is an (n, k)-*universal set* if for every k-subset W of Z_n and for any partition (W_0, W_1) of W, there is a splitting function f in $\Psi_{n,k}$ that implements (W_0, W_1). The *size* of an (n, k)-universal set $\Psi_{n,k}$ is the number of splitting functions in $\Psi_{n,k}$.

Proposition 1.1 (*[13,2]*). *There is an $O(n2^{k+12\log^2 k})$ time deterministic algorithm that constructs an (n, k)-universal set $\Psi_{n,k}$ of size bounded by $n2^{k+12\log^2 k+2}$.*

A function f on Z_n is *injective* from a subset W of Z_n if for any two different elements x and y in W, $f(x) \neq f(y)$.

By Bertrand's postulate, proved by Chebyshev in 1850 (see [14], Section 5.2), there is a prime number q such that $n \le q < 2n$. Moreover, the smallest prime number q_0 between n and $2n$ can be constructed in time $O(n)$.

Proposition 1.2 (*[6]*). *Let n and k be integers, $n \ge k$, and let q_0 be the smallest prime number such that $n \le q_0 < 2n$. For any k-subset W in Z_n, there is an integer z, $0 \le z < q_0$, such that the function $g_{n,k,z}$ over Z_n, defined as $g_{n,k,z}(a) = (az \bmod q_0) \bmod k^2$, is injective from W.*

2. An improved algorithm for weighted rD-MATCHING

Let $S \subseteq A_1 \times \cdots \times A_r$ be a collection of r-tuples. Denote by $\mathrm{Val}^i(S)$ the set of all elements from A_i in S, $1 \le i \le r$, and let $\mathrm{Val}(S) = \bigcup_{i=1}^r \mathrm{Val}^i(S)$. Without loss of generality, we assume that $|\mathrm{Val}^i(S)| = n$ for all i. A *matching* is a collection of r-tuples in which no two tuples have common elements. A *k-matching* is a matching of k tuples.

Our improved algorithm for weighted rD-MATCHING is based on the idea of *divide-and-conquer*. Let (S, k) be an instance of weighted rD-MATCHING. Suppose that a k-matching of the maximum weight in S is M^*. Let M_1^* be the collection of any $k/2$ r-tuples in M^*, and let M_2^* be the collection of the other $k/2$ r-tuples in M^*.[2] Then both M_1^* and M_2^* are $(k/2)$-matchings. If we can partition the elements in $\mathrm{Val}(S)$ into two subsets V_1 and V_2 such that $\mathrm{Val}(M_1^*) \subseteq V_1$ and $\mathrm{Val}(M_2^*) \subseteq V_2$, then we can recursively work on two subcollections S_1 and S_2, where S_i is the subcollection of r-tuples in S in which all elements are in V_i, for $i = 1, 2$. Note that for $i = 1, 2$, M_i^* is a $(k/2)$-matching in S_i. Therefore, if we can recursively find a $(k/2)$-matching M_i' of the maximum weight in S_i, for $i = 1, 2$, then, since $\mathrm{Val}(S_1) \cup \mathrm{Val}(S_2) = \emptyset$, the union of M_1' and M_2' is a k-matching of the maximum weight in S.

This idea has been used in [2], which developed a deterministic algorithm of time $O^*(4^{rk+o(k)})$ for the weighted rD-MATCHING problem. The following observation enables us to further improve the running time of the algorithm. Let $\mathrm{Val}^1(S) = \{a_1, \ldots, a_n\}$. Then it is easy to see that there is an index h such that $\{a_1, \ldots, a_h\}$ contains $k/2$ elements in $\mathrm{Val}^1(M^*)$ and $\{a_{h+1}, \ldots, a_n\}$ contains the other $k/2$ elements in $\mathrm{Val}^1(M^*)$. In particular, this implies that the maximum weighted k-matching M^* can be partitioned into two $(k/2)$-matchings M_1^* and M_2^* such that $\{a_1, \ldots, a_h\}$ contains all elements in $\mathrm{Val}^1(M_1^*)$ and $\{a_{h+1}, \ldots, a_n\}$ contains all elements in $\mathrm{Val}^1(M_2^*)$. The index h can be found by enumerating all possible indices between 1 and n. Therefore, finding the correct partition of the elements in $\mathrm{Val}^1(S)$ takes at most n rounds.

This idea is implemented in the algorithm **WRDM** given in Fig. 1, where the function $g_{n,k,z}$ is as given in Proposition 1.2.

Theorem 2.1. *The algorithm* **WRDM** *in Fig. 1 correctly solves the weighted rD-MATCHING problem in time $O^*(4^{(r-1)k+o(k)})$.*

Proof. First note that the collection M returned by the algorithm **WRDM** is initialized as the empty set $\emptyset$ in step 4. Only when step 5.1 of the algorithm finds a k-matching M_1 in S, does step 5.2 of the algorithm replace M by the k-matching M_1. Therefore, if the collection S has no k-matching, then the algorithm **WRDM** will always correctly return the empty set $\emptyset$.

In consequence, we only need to prove that when the collection S contains k-matchings, the algorithm **WRDM**(S, k) must return a k-matching of the maximum weight in S.

Without loss of generality and by renaming the elements, we can assume that the set $\mathrm{Val}^1(S)$ is the set Z_n, and that the set $\mathrm{Val}(S) - \mathrm{Val}^1(S)$ is the set $Z_{(r-1)n}$. In particular, for an h-matching M in S for any integer h, $\mathrm{Val}^1(M)$ is a subset of h elements in Z_n and $\mathrm{Val}(M) - \mathrm{Val}^1(M)$ is a subset of $(r-1)h$ elements in $Z_{(r-1)n}$.

We prove the following Claim for the subroutine **Matching-ext**(S', z_1, z_2, h) by induction on the integer h.

[2] To simplify our discussion, we assume that k is even. In case k is odd, we should replace the two $k/2$'s by $\lfloor k/2 \rfloor$ and $\lceil k/2 \rceil$, respectively. Our discussion will still go through but will be involved in more complicated expressions.

```
Algorithm WRDM ($S$, $k$)
Input: $S \subseteq A_1 \times A_2 \times \cdots \times A_r$ and an integer $k$
Output: a maximum weighted $k$-matching in $S$ if such a matching exists
1.   for $h = 1$ to $k$ do construct a $(((r-1)k)^2, (r-1)h)$-universal set $\Psi_{((r-1)k)^2,(r-1)h}$;
2.   let $q_1$ be the smallest prime number such that $n \le q_1 < 2n$;
3.   let $q_2$ be the smallest prime number such that $(r-1)n \le q_2 < 2(r-1)n$;
4.   $M = \emptyset$;
5.   for $0 \le z_1 \le q_1$ and $0 \le z_2 \le q_2$ do
5.1      $M_1$ = Matching-ext($S, z_1, z_2, k$);
5.2      if $M_1 \ne \emptyset$ and $M_1$ is a $k$-matching with weight larger than that of $M$
         then $M = M_1$;
6.   return $M$.

Subroutine Matching-ext($S', z_1, z_2, h$)
Input: a collection $S'$ of $r$-tuples and an integer $h \le k$, $z_1$ induces a pre-partition of
          $\mathrm{Val}^1(S')$, and $z_2$ induces a pre-partition of $\mathrm{Val}(S') - \mathrm{Val}^1(S')$.
Output: an $h$-matching with maximum weight in $S'$ if such a matching exists
1. if $h = 1$ then return the $r$-tuple with the maximum weight in $S'$;
2. $M' = \emptyset$;
3. for $i = 0$ to $k^2 - 1$ do
       for each splitting function $f$ in $\Psi_{((r-1)k)^2,(r-1)h}$ do
3.1    $V_0 = \{a \mid a \in \mathrm{Val}^1(S')$ and $g_{n,h,z_1}(a) \le i\}$;
3.2    $V_1 = \{a \mid a \in \mathrm{Val}^1(S')$ and $g_{n,h,z_1}(a) > i\}$;
3.3    $W_0 = \{a \mid a \in \mathrm{Val}(S') - \mathrm{Val}^1(S')$ and $f(g_{(r-1)n,(r-1)k,z_2}(a)) = 0\}$;
3.4    $W_1 = \{a \mid a \in \mathrm{Val}(S') - \mathrm{Val}^1(S')$ and $f(g_{(r-1)n,(r-1)k,z_2}(a)) = 1\}$;
3.5    let $S'_0$ be the subcollection of $r$-tuples in $S'$ whose elements are all in $V_0 \cup W_0$;
3.6    let $S'_1$ be the subcollection of $r$-tuples in $S'$ whose elements are all in $V_1 \cup W_1$;
3.7    $M'_0$ = Matching-ext($S'_0, z_1, z_2, h/2$);
3.8    $M'_1$ = Matching-ext($S'_1, z_1, z_2, h/2$);
3.9    if $M'_0 \ne \emptyset$, $M'_1 \ne \emptyset$, and weight($M'_0$) + weight($M'_1$) > weight($M'$)
       then $M' = M'_0 \cup M'_1$;
4.   return $M'$.
```

Fig. 1. The algorithm **WRDM**.

Claim. Let M^* be an h-matching of the maximum weight in the collection S', where $S' \subseteq S$ and $h \le k$. If z_1 is an integer that makes the function g_{n,k,z_1} injective from $\mathrm{Val}^1(M^*)$, and if z_2 is an integer that makes the function $g_{(r-1)n,(r-1)k,z_2}$ injective from $\mathrm{Val}(M^*) - \mathrm{Val}^1(M^*)$, then the subroutine **Matching-ext**(S', z_1, z_2, h) returns an h-matching of the maximum weight in the collection S'.

The Claim obviously holds true for the case $h = 1$ by step 1 of the subroutine (for any given z_1 and z_2). Now we consider the case $h > 1$. Recall that the function g_{n,k,z_1} is from Z_n to Z_{k^2}. Since the function g_{n,k,z_1} is injective from $\mathrm{Val}^1(M^*)$, we can assume that g_{n,k,z_1} maps the h elements in $\mathrm{Val}^1(M^*)$ to h different elements $i_1, i_2, \ldots, i_h$ in Z_{k^2}, where $0 \le i_1 < i_2 < \cdots < i_h \le k^2 - 1$. Take the index $i_{h/2}$ and define

$$M^*_0 : \text{the set of } h/2\ r\text{-tuples in } M^* \text{ such that} \quad \forall a \in \mathrm{Val}^1(M^*_0), g_{n,h,z_1}(a) \le i_{h/2}, \tag{1}$$

$$M^*_1 : \text{the rest } h/2\ r\text{-tuples in } M^* \text{ such that} \quad \forall b \in \mathrm{Val}^1(M^*_1), g_{n,h,z_1}(b) > i_{h/2}. \tag{2}$$

Now consider the set $\mathrm{Val}(M^*) - \mathrm{Val}^1(M^*)$. First of all, by our assumption, the function $g_{(r-1)n,(r-1)k,z_2}$, which is from $Z_{(r-1)n}$ to $Z_{((r-1)k)^2}$, is injective from $\mathrm{Val}(M^*) - \mathrm{Val}^1(M^*)$. Therefore, the function $g_{(r-1)n,(r-1)k,z_2}$ maps the set $\mathrm{Val}(M^*) - \mathrm{Val}^1(M^*)$ of $(r-1)h$ elements to a set X of $(r-1)h$ different elements in $Z_{((r-1)k)^2}$. In particular, the function $g_{(r-1)n,(r-1)k,z_2}$ maps the set $\mathrm{Val}(M^*_0) - \mathrm{Val}^1(M^*_0)$ of $(r-1)h/2$ elements to a set X_0 of $(r-1)h/2$ different elements in $Z_{((r-1)k)^2}$, and maps the set $\mathrm{Val}(M^*_1) - \mathrm{Val}^1(M^*_1)$ of $(r-1)h/2$ elements to a set X_1 of $(r-1)h/2$ different elements in $Z_{((r-1)k)^2}$. That is

$$g_{(r-1)n,(r-1)k,z_2} \text{ maps } \mathrm{Val}(M^*_0) - \mathrm{Val}^1(M^*_0) \text{ to } X_0, \tag{3}$$

$$g_{(r-1)n,(r-1)k,z_2} \text{ maps } \mathrm{Val}(M^*_1) - \mathrm{Val}^1(M^*_1) \text{ to } X_1. \tag{4}$$

Note that (X_0, X_1) makes a partition of X (i.e., $X_0 \cap X_1 = \emptyset$ and $X_0 \cup X_1 = X$).

Since X is a subset of $(r-1)h$ elements in $Z_{((r-1)k)^2}$, by the definition of the $(((r-1)k)^2, (r-1)h)$-universal set $\Psi_{((r-1)k)^2,(r-1)h}$, there is a splitting function f_0 in $\Psi_{((r-1)k)^2,(r-1)h}$ that implements the partition (X_0, X_1), that is

$$f_0(a) = 0 \text{ for all } a \in X_0, \tag{5}$$

$$f_0(b) = 1 \text{ for all } b \in X_1. \tag{6}$$

Now consider step 3 of the subroutine **Matching-ext**(S', z_1, z_2, h), when the integer $i = i_{h/2}$ is picked and the splitting function $f = f_0$ is picked. For these selections of the index $i = i_{h/2}$ and the function $f = f_0$, we can derive

$$\mathrm{Val}^1(M_0^*) \subseteq V_0 \quad \text{and} \quad \mathrm{Val}^1(M_1^*) \subseteq V_1, \tag{7}$$

where the first relation is from (1) and step 3.1 of the subroutine, and the second relation is from (2) and step 3.2 of the subroutine. Moreover, from (3), (5), and step 3.3 of the subroutine, we derive

$$\mathrm{Val}(M_0^*) - \mathrm{Val}^1(M_0^*) \subseteq W_0, \tag{8}$$

and from (4), (6), and step 3.4 of the subroutine, we derive

$$\mathrm{Val}(M_1^*) - \mathrm{Val}^1(M_1^*) \subseteq W_1. \tag{9}$$

From (7) and (8), all elements in M_0^* are contained in $V_0 \cup W_0$, and from (7) and (9), all elements in M_1^* are contained in $V_1 \cup W_1$. By steps 3.5–3.6, the subcollection S_0' contains the $(h/2)$-matching M_0^*, and the subcollection S_1' contains the $(h/2)$-matching M_1^*. Note that M_0^* must be an $(h/2)$-matching of the maximum weight in S_0'—otherwise, a maximum weighted $(h/2)$-matching in S_0' plus the $(h/2)$-matching M_1^* in S_1' would form an h-matching whose weight is larger than that of M^*, contradicting the maximality of M^*. Since the function g_{n,k,z_1} is injective from $\mathrm{Val}^1(M^*)$, the integer z_1 also makes the function g_{n,k,z_1} injective from $\mathrm{Val}^1(M_0^*)$. Similarly, the integer z_2 makes the function $g_{(r-1)n,(r-1)k,z_2}$ injective from $\mathrm{Val}(M_0^*) - \mathrm{Val}^1(M_0^*)$. Therefore, by the induction hypothesis, the subroutine call **Matching-ext**$(S_0', z_1, z_2, h/2)$ in step 3.7 will return an $(h/2)$-matching M_0' of the maximum weight in S_0', where the $(h/2)$-matching M_0' should have the same weight as that of M_0^*. Completely similar analysis shows that the subroutine call **Matching-ext**$(S_1', z_1, z_2, h/2)$ in step 3.8 will return an $(h/2)$-matching M_1' of the maximum weight in S_1', where the $(h/2)$-matching M_1' should have the same weight as that of M_1^*. Since the collections S_0' and S_1' share no common elements, the union of M_0' and M_1' is an h-matching in S' whose weight is equal to that of the maximum weighted h-matching M^*, we conclude that after step 3.9 of the subroutine for the selections of the index $i = i_{h/2}$ and the splitting function $f = f_0$, the collection M' becomes an h-matching of the maximum weight in S'. In particular, when the subroutine **Matching-ext**(S', z_1, z_2, h) returns at step 4, it returns an h-matching of the maximum weight in S'.

This completes the proof of the Claim.

Now we return back to the algorithm **WRDM**(S, k). Suppose that the collection S has a k-matching $\bar{M}$ of the maximum weight. Since the set $\mathrm{Val}^1(\bar{M})$ is a subset of k elements in $\mathrm{Val}^1(S) = Z_n$, by Proposition 1.2, there is an integer z_1, $0 \le z_1 \le q_1$, such that the function g_{n,k,z_1} is injective from $\mathrm{Val}^1(\bar{M})$. Similarly, since the set $\mathrm{Val}(\bar{M}) - \mathrm{Val}^1(\bar{M})$ is a subset of $(r-1)k$ elements in $\mathrm{Val}(S) - \mathrm{Val}^1(S) = Z_{(r-1)n}$, by Proposition 1.2, there is an integer z_2, $0 \le z_2 \le q_2$, such that the function $g_{(r-1)n,(r-1)k,z_2}$ is injective from $\mathrm{Val}(\bar{M}) - \mathrm{Val}^1(\bar{M})$. When these values of z_1 and z_2 are selected in step 5 of the algorithm, by the Claim proved above, the subroutine call **Matching-ext**(S, z_1, z_2, k) in step 5.1 will return a k-matching of the maximum weight in S. Now by step 5.2 of the algorithm, the final collection M returned by the algorithm in step 6 is a k-matching of the maximum weight in S.

This completes the proof of the correctness for the algorithm **WRDM**.

Finally, we study the complexity of the algorithm **WRDM**. First consider the subroutine **Matching-ext**(S', z_1, z_2, h). By Proposition 1.1, the $(((r-1)k)^2, (r-1)h)$-universal set $\Psi_{((r-1)k)^2,(r-1)h}$ contains at most $((r-1)k)^2 2^{(r-1)h + 12\log^2((r-1)h)+2}$ splitting functions. Therefore, the loop body, i.e., steps 3.1–3.9, of the subroutine is executed at most

$$\begin{aligned} k^2((r-1)k)^2 2^{(r-1)h+12\log^2((r-1)h)+2} &\le 2^{(r-1)h} k^2((r-1)k)^2 2^{12\log^2((r-1)k)+2} \\ &= 2^{(r-1)h} k^2 2^{12\log^2((r-1)k)+2\log((r-1)k)+2} \\ &\le 2^{(r-1)h} 2^{12\log^2((r-1)k)+4\log((r-1)k)+2} \end{aligned}$$

times (note that $r \ge 3$). Each execution of the loop body takes time $O(m)$ to construct the subcollections S_0' and S_1', and recursively calls the subroutine twice to search for $(h/2)$-matchings in the subcollections, where $m = O(n^r)$ is the size of the collection S'. Therefore, if we let $T(m, h)$ be the running time of the subroutine **Matching-ext**(S', z_1, z_2, h), then the function $T(m, h)$ satisfies the following recurrence relation:

$$\begin{aligned} T(m, 1) &\le cm; \\ T(m, h) &\le 2^{(r-1)h} 2^{12\log^2((r-1)k)+4\log((r-1)k)+2} \cdot (2T(m, h/2) + cm), \end{aligned}$$

where c is a constant. Using Corollary 2.2 in [2] (where we replace k by h, n by m, a by 2^{r-1}, $t(n)$ by cm, and c_0 by $2^{12\log^2((r-1)k)+4\log((r-1)k)+2}$), we derive

$$T(m, h) = O(m 4^{(r-1)h} 2^{O(\log^3((r-1)k))}) = O^*(4^{(r-1)h+o(k)}).$$

This shows that the running time of the subroutine **Matching-ext**(S', z_1, z_2, h) is bounded by $O^*(4^{(r-1)h+o(k)})$.

Since the prime number q_1 is bounded by $2n$, and the prime number q_2 is bounded by $2(r-1)n$, and by Proposition 1.1, the $(((r-1)k)^2, (r-1)h)$-universal sets for all h in step 1 can be constructed in time $O^*(2^{k+o(k)})$, from the analysis for the subroutine **Matching-ext**(S', z_1, z_2, h), now it is straightforward to conclude that the running time of the algorithm **WRDM**(S, k) is bounded by $O^*(4^{(r-1)k+o(k)})$.

This completes the proof of the theorem. □

We point out that the algorithm **WRDM**(S, k) can be used directly to solve the unweighted rD-MATCHING problem in time $O^*(4^{(r-1)k+o(k)})$, which improves the previous best deterministic algorithm of running time $O^*(4^{rk+o(k)})$ for the problem [2]. Applying the algorithm **WRDM**(S, k) to the unweighted 3D-MATCHING problem, we get a deterministic algorithm of running time $O^*(16^{k+o(k)})$ for the problem, improving the previous best deterministic algorithm of running time $O^*(21.26^k)$ for the problem [12].

3. An improved algorithm for weighted *r*-SET PACKING

In this section, we are focused on the weighted r-SET PACKING problem. A *packing* of r-sets is a collection of r-sets in which no two r-sets intersect. A *k-packing* is a packing of k r-sets. Let (S, k) be an instance of weighted r-SET PACKING, whose objective is to find a k-packing with the maximum weight in S. For a collection S' of r-sets, let $\mathrm{Val}(S')$ be the set of all elements occurring in S'. Without loss of generality, let $\mathrm{Val}(S) = Z_n = \{0, 1, \ldots, n-1\}$ so that the elements in $\mathrm{Val}(S)$ can be related by inequalities.

The method we used in the previous section on rD-MATCHING is not applicable to r-SET PACKING: the elements in $\mathrm{Val}(S)$ cannot be partitioned into "columns". Nevertheless, the following observation helps us to "partially" apply the method.

Define the *pivot* of an r-set σ to be the smallest element in σ.

Lemma 3.1. *Let $P = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ be a k-packing of the maximum weight in the collection S, sorted in increasing order by the pivots of the r-sets. For each i, $1 \le i \le k$, let a_i be the pivot of the r-set σ_i. Then for all i, $1 \le i \le k$, no r-set among $\{\sigma_{i+1}, \ldots, \sigma_k\}$ can contain any elements in $\{a_1, \ldots, a_i\}$.*

Proof. Since the r-sets $\sigma_1, \sigma_2, \ldots, \sigma_k$ are sorted in increasing order by their pivots, for each j, $j > i$, the smallest element in σ_j (i.e., the pivot of σ_j) is larger than all elements in $\{a_1, \ldots, a_i\}$. Thus, the r-set σ_j cannot contain any elements in $\{a_1, \ldots, a_i\}$. □

In particular, if we pick the pivot $a_{k/2}$ of the r-set $\sigma_{k/2}$ in the maximum weighted k-packing $P = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$, which is sorted in increasing order by pivots,[3] then all r-sets in the $(k/2)$-packing $P_0 = \{\sigma_1, \ldots, \sigma_{k/2}\}$ have their pivots smaller than or equal to $a_{k/2}$, and all r-sets in the $(k/2)$-packing $P_1 = \{\sigma_{k/2+1}, \ldots, \sigma_k\}$ have their pivots larger than $a_{k/2}$. Unfortunately, an r-set σ_i in P_0, where $i \le k/2$, may contain an element a that is a pivot for some r-set not in P and $a > a_{k/2}$. Therefore, we cannot simply partition the collection S into two subcollections S_0 and S_1 based on pivots, then construct maximum weighted $(k/2)$-packings in S_0 and S_1, and combine them into a maximum weighted packing in S.

On the other hand, the element $a_{k/2}$ at least indicates the fact that if we want to construct two disjoint subcollections S_0 and S_1 from S such that P_0 and P_1 are maximum weighted $(k/2)$-packings in S_0 and S_1, respectively, then by Lemma 3.1, all pivots $a_1, \ldots, a_{k/2}$ should be in S_0. Therefore, we only need to concentrate on properly partitioning the rest $(r-1/2)k$ elements in the k-packing P into the two subcollections S_0 and S_1.

Let f_0 be a function on $\mathrm{Val}(S)$. Define $f_0(S)$ to be a collection of r-sets constructed by the following process. Initially, $f_0(S) = \emptyset$.

1. For each r-set $\sigma = \{b_1, b_2, \ldots, b_r\}$ in S, if $f_0(b_i) \neq f_0(b_j)$ for all $i \neq j$, construct an r-set $f_0(\sigma) = \{f_0(b_1), f_0(b_2), \ldots, f_0(b_r)\}$ of the same weight and add it to $f_0(S)$;
2. If there are multiple copies of an r-set σ' in $f_0(S)$, then remove all copies of σ' from $f_0(S)$ except the one with the largest weight.

For a k-packing $P' = \{\sigma'_1, \ldots, \sigma'_k\}$ in $f_0(S)$, there is a k-packing $P = \{\sigma_1, \ldots, \sigma_k\}$ of the same weight in S, where for each i, $f_0(\sigma_i) = \sigma'_i$. Moreover, the k-packing P can be easily constructed from P' in time $O(km)$, where m is the total number of r-sets in S. In particular, the weight of a maximum weighted k-packing in $f_0(S)$ cannot be larger than that of a maximum weighted k-packing in S.

Now we are ready for our algorithm **WRSP** for weighted r-SET PACKING, as given in Fig. 2.

Theorem 3.2. *The algorithm* **WRSP** *correctly solves the weighted r-SET PACKING problem in time $O^*(2^{(2r-1)k+o(k)})$.*

Proof. The function $g_{n,rk,z}$ in step 4.1 of the algorithm **WRSP** is as given in Proposition 1.2, which maps each element in Z_n to an element in $Z_{(rk)^2}$. First note that in the algorithm **WRSP**, the collection $\overline{P}$ is initialized to the empty set $\emptyset$ in step 3. The collection $\overline{P}$ becomes a k-packing in $\overline{S}$ only when step 4.3 of the algorithm finds a k-packing in $\overline{S}$. Moreover, by the remark before the theorem, when $\overline{P}$ is a k-packing in $\overline{S}$, a k-packing P in S of the same weight can be constructed in time $O(km)$,

[3] Again, to simplify our discussion and expressions, we assume that k is an even number.

```
Algorithm WRSP (S, k)
Input: a collection S of r-sets such that Val(S) = Z_n, and an integer k
Output: a maximum weighted k-packing in S if such a packing exists
1.   for $h = 1$ to $k$ do construct a $((rk)^2, (2r-1)h/2)$-universal set $\Psi_{(rk)^2,(2r-1)h/2)}$;
2.   let $q$ be the smallest prime number such that $n \le q < 2n$;
3.   $\overline{P} = \emptyset$;
4.   for $z = 0$ to $q - 1$ do
4.1      $\overline{S} = g_{n,rk,z}(S)$;
4.2      $\overline{P}_1 = $ Packing-ext($\overline{S}$, $k$);
4.3      if $\overline{P}_1$ is a $k$-packing with weight larger than that of $\overline{P}$ then $\overline{P} = \overline{P}_1$;
5.   return a $k$-packing $P$ in $S$ whose weight is equal to that of $\overline{P}$.

Subroutine Packing-ext($S'$, $h$)
Input: a collection $S'$ of $r$-sets such that $\text{Val}(S') \subseteq Z_{(rk)^2}$ and an integer $h \le k$
Output: a maximum weighted $h$-packing in $S'$ if such a packing exists
1.   if $h = 1$ then return the $r$-set with the maximum weight in $S'$;
2.   let $\{a_1, a_2, \ldots, a_t\}$ be the set of pivots of $r$-sets in $S'$, sorted in increasing order;
3.   $P' = \emptyset$;
4.   for $i = h/2$ to $t - h/2$ do
4.1      for each splitting function $f$ in $\Psi_{(rk)^2,(2r-1)h/2}$ do
4.2        $V_0 = \{a_1, a_2, \ldots, a_i\}$;
4.3        $W_0 = \{x \mid x \in \text{Val}(S') - V_0 \text{ and } f(x) = 0\}$;
4.4        $W_1 = \{x \mid x \in \text{Val}(S') - V_0 \text{ and } f(x) = 1\}$;
4.5        let $S'_0$ be the subcollection of $S'$ that contains only elements in $W_0 \cup V_0$;
4.6        let $S'_1$ be the subcollection of $S'$ that contains only elements in $W_1$;
4.7        $P'_0 = $ Packing-ext($S'_0$, $h/2$);     $P'_1 = $ Packing-ext($S'_1$, $h/2$);
4.8        if $P'_0 \ne \emptyset, P'_1 \ne \emptyset$ and weight($P'_0$) + weight($P_1$) > weight($P'$)
           then $P' = P'_0 \cup P'_1$;
5.   return $P'$.
```

Fig. 2. The algorithm **WRSP**.

where m is the total number of r-sets in S. Therefore, if the collection S has no k-packing, then the algorithm **WRSP** will always correctly return the empty set $\emptyset$.

As a result, we only need to prove that when S contains k-packings, the algorithm **WRSP** must return a k-packing with the maximum weight in S. For this, we first prove that if a collection S' contains an h-packing, then the subroutine **Packing-ext**(S', h) returns an h-packing of the maximum weight in S'. Our proof is by induction on h.

The Claim obviously holds true for the case $h = 1$ by step 1 of the subroutine. Now consider the case when $h > 1$. Let $P^* = \{\sigma_1, \sigma_2, \ldots, \sigma_h\}$ be an h-packing of the maximum weight in S', where the r-sets are sorted in increasing order by their pivots, and for each i, let b_i be the pivot of σ_i. Note that all pivots b_i appear in the set $\{a_1, a_2, \ldots, a_t\}$ constructed in step 2 of the subroutine. In particular, we can assume $b_{h/2} = a_j$.

Let $P_0^* = \{\sigma_1, \ldots, \sigma_{h/2}\}$, and $P_1^* = \{\sigma_{h/2+1}, \ldots, \sigma_h\}$. Consider the two disjoint sets $Y_0 = \text{Val}(P_0^*) - \{a_1, \ldots, a_j\}$ and $Y_1 = \text{Val}(P_1^*)$. The set Y_0 has at most $(r-1)h/2$ elements because $\{b_1, \ldots, b_{h/2}\}$ is a subset of $\{a_1, \ldots, a_j\}$ (note that some a_i in $\{a_1, \ldots, a_j\}$ that is not a pivot of any r-set in P_0^* can appear in P_0^*). The set Y_1 has exact $(rh)/2$ elements. Let Y_0' be any superset of Y_0 that is disjoint from Y_1 and has exact $(r-1)h/2$ elements. Then, the set $Y = Y_0' \cup Y_1$ is a set of exact $(2r-1)h/2$ elements in the set $Z_{(rk)^2}$ (recall that we assume $\text{Val}(S') \subseteq Z_{(rk)^2}$).

By the definition of the $((rk)^2, (2r-1)h/2)$-universal set, there is a splitting function f^* in $\Psi_{(rk)^2,(2r-1)h/2}$ such that (1) $f^*(x) = 0$ for all x in Y_0' (in particular $f^*(x) = 0$ for all x in Y_0), and (2) $f^*(x) = 1$ for all x in Y_1. Now consider step 4 of the subroutine when $i = j$ (i.e., when $a_i = b_{h/2}$) and when $f = f^*$. By the above discussion, we have $\text{Val}(P_0^*) \subseteq W_0 \cup V_0$ and $\text{Val}(P_1^*) \subseteq W_1$ (note that by Lemma 3.1, no element in $\{a_1, \ldots, a_j\}$ is in $\text{Val}(P_1^*)$). Therefore, P_0^* and P_1^* are $(h/2)$-packings in the subcollections S_0' and S_1', respectively. By the induction hypothesis, step 4.7 returns a maximum weighted $(h/2)$-packing P_0' in S_0' and a maximum weighted $(h/2)$-packing P_1' in S_1'. Since the subcollections S_0' and S_1' contain no common elements, the union $P_0' \cup P_1'$ is an h-packing in S'. Since the weight of P_0' is not smaller than that of P_0^* and the weight of P_1' is not smaller than that of P_1^*, and $P^* = P_0^* \cup P_1^*$ is a maximum weighted h-packing in S', we conclude that after the execution of the loop with the values $i = j$ and $f = f^*$, the collection P' becomes a maximum weighted h-packing in S'. Since step 4.8 of the subroutine can never produce an h-packing of weight larger than that of P^*, the subroutine **Packing-ext**(S', h) must return in step 5 a maximum weighted h-packing in S'. This completes the proof that if S' contains h-packings, then the subroutine **Packing-ext**(S', h) must return a maximum weighted h-packing in S'.

To complete the proof for the correctness for the algorithm **WRSP**(S, k), let $P_{\max}$ be a maximum weighted k-packing in S, and let $Y_{\max}$ be the set of rk elements in $P_{\max}$. By Proposition 1.2, there is an integer z^*, $0 \le z^* < q$, such that the function g_{n,rk,z^*} is injective from $Y_{\max}$. Therefore, when $z = z^*$ in step 4 of the algorithm **WRSP**, the collection $g_{n,rk,z^*}(P_{\max})$ has the same weight as that of $P_{\max}$ and is a maximum weighted k-packing in $\overline{S}$ (note that the weight of a maximum weighted k-packing in $\overline{S}$ cannot be larger than that of $P_{\max}$). By what we have proved for the subroutine **Packing-ext**, step 4.2 of the algorithm will produce a maximum weighted k-packing $\overline{P}_1$ in $\overline{S}$ and step 4.3 will ensure that the collection $\overline{P}$ is a k-packing in $\overline{S}$ whose weight is equal to that of $P_{\max}$. Finally, step 5 of **WRSP** returns a maximum weighted k-packing in S.

We now analyze the complexity of the algorithm. First consider the subroutine **Packing-ext**. By our assumption, $\mathrm{Val}(S') \subseteq Z_{(rk)^2}$. Thus, the set of pivots in step 2 of the subroutine has at most $(rk)^2$ elements (i.e., $t \leq (rk)^2$). By Proposition 1.1, the $((rk)^2, (2r-1)h/2)$-universal set $\Psi_{(rk)^2,(2r-1)h/2}$ has at most $(rk)^2 2^{(2r-1)h/2+12\log^2((2r-1)h/2)+2}$ splitting functions. Therefore, the loop of steps 4.2–4.8 of the subroutine is executed at most $(rk)^4 2^{(2r-1)h/2+12\log^2((2r-1)h/2)+2}$ times. Each execution of the loop, not counting the recursive calls, takes time $O(m)$, where m is the number of r-sets in S'.

Let $T(m, h)$ be the running time of the subroutine **Packing-ext**. Then we have

$$\begin{aligned} T(m,1) &\leq cm; \\ T(m,h) &\leq (rk)^4 2^{(2r-1)h/2+12\log^2((2r-1)h/2)+2}(cm + 2T(m, h/2)) \\ &\leq 2^{(2r-1)h/2} 2^{12\log^2(rk)+4\log(rk)+2}(cm + 2T(m, h/2)) \end{aligned}$$

where c is a constant, and we have used the fact $h \leq k$ and $(2r-1)/2 < r$. Using Corollary 2.2 in [2], where we replace k by h, n by m, a by $2^{(2r-1)/2}$, $t(n)$ by cm, and c_0 by $2^{12\log^2(rk)+4\log(rk)+2}$ (note that both r and k are independent of h and m), we derive

$$T(m,h) = O(m 2^{(2r-1)h} 2^{O(\log^3(rk))}) = O^*(2^{(2r-1)h+o(k)}).$$

This shows that the complexity of the subroutine **Packing-ext**(S', h) is $O^*(2^{(2r-1)h+o(k)})$. In particular, step 4.2 of the algorithm **WRSP** takes time $O^*(2^{(2r-1)k+o(k)})$.

By Proposition 1.1, the $((rk)^2, (2r-1)h/2)$-universal sets $\Psi_{(rk)^2,(2r-1)h/2}$ for all h in step 1 of the algorithm **WRSP** can be constructed in time $O^*(2^{(2r-1)k/2+o(k)})$. The prime number q is bounded by $2n$. Therefore, the running time of the algorithm **WRSP**(S, k) is bounded by $O^*(2^{(2r-1)k+o(k)})$. This completes the proof of theorem. □

We point out that the algorithm **WRSP**(S, k) can be directly applied to solve the unweighted r-SET PACKING problem in time $O^*(2^{(2r-1)k+o(k)})$, and improves the previous best result $O^*(2^{2rk+o(k)})$ [2]. In particular, when applying the algorithm **WRSP**(S, k) to the unweighted 3-SET PACKING problem, we solve unweighted 3-SET PACKING in time $O^*(32^{k+o(k)})$, improving the previous best result $O^*(43.62^{k+o(k)})$ [16].

4. Kernelization for weighted matching and packing

In this section, we present an $O^*(k^r)$ kernel for the weighted r-SET PACKING problem (thus, also for the weighted rD-MATCHING problem).

Lemma 4.1. *Let (S, k) be an instance of weighted r-SET PACKING, where S contains k-packings. Let $a_1, \ldots, a_{r-q}$ be $r-q$ elements in $\mathrm{Val}(S)$, $0 \leq q \leq r-1$. If S has more than $(rk)^q$ r-sets that contain all $a_1, \ldots, a_{r-q}$, then there is a proper subcollection S' of S, $S' \subsetneq S$, that contains a maximum weighted k-packing in S.*

Proof. We prove the lemma by induction on q. The lemma is trivially true for the case $q = 0$: if the collection S contains more than $(rk)^0 = 1$ copies of an r-set that consists of the r elements $a_1, \ldots, a_r$, then we can remove all but the one with the largest weight of these copies. It is obvious that the resulting proper subcollection S' of S contains a maximum weighted k-packing in S.

Now consider the general case of $q > 0$. Let $T = \{\sigma_1, \ldots, \sigma_h, \sigma_{h+1}, \ldots, \sigma_t\}$ be the collection of all r-sets in S that contain all elements $a_1, \ldots, a_{r-q}$, sorted by their weights in non-increasing order, where $h = (rk)^q$ and $t > h = (rk)^q$. Let P^* be a maximum weighted k-packing in S.

Case 1. If P^* has no r-set that contains all elements $a_1, \ldots, a_{r-q}$, or if P^* contains an r-set σ_i in T with $i \leq h$, then the proper subcollection $S' = S - \{\sigma_t\}$ of S contains P^* (note that $h < t$ and that no other r-set in P^* can be in T), and the lemma is proved.

The remaining case is that P^* contains an r-set σ_i in T with $i > h$. We divide this case into two subcases.

Case 2.1. If an r-set σ_j in T with $j \leq h$ contains no element in $\mathrm{Val}(P^*) - \{a_1, \ldots, a_{r-q}\}$, then $(P^* - \{\sigma_i\}) \cup \{\sigma_j\}$ is a maximum weighted k-packing in S that is contained in the proper subcollection $S' = S - \{\sigma_t\}$ (note that by the ordering of T and by the definition of P^*, the weight of σ_j must be equal to the weight of σ_i) and the lemma is proved.

Case 2.2. Every r-set in $\{\sigma_1, \ldots, \sigma_h\}$ contains at least one element in $\mathrm{Val}(P^*) - \{a_1, \ldots, a_{r-q}\}$. Then $\{\sigma_1, \ldots, \sigma_h, \sigma_i\}$ is a collection of more than $h = (rk)^q$ r-sets in T that contain all elements $a_1, \ldots, a_{r-q}$ plus at least one element in $\mathrm{Val}(P^*) - \{a_1, \ldots, a_{r-q}\}$. Since there are $r(k-1) + q < rk$ elements in $\mathrm{Val}(P^*) - \{a_1, \ldots, a_{r-q}\}$ (note that $q < r$), there is at least one element b in $\mathrm{Val}(P^*) - \{a_1, \ldots, a_{r-q}\}$ such that more than $(rk)^{q-1}$ r-sets in T (thus in S) contain the $r-(q-1)$ elements $\{a_1, \ldots, a_{r-q}, b\}$. Now by the induction hypothesis, there is a proper subcollection S' of S that contains a maximum weighted k-packing in S.

This completes the proof of the lemma. □

Now we can conclude with our main theorem in this section.

Theorem 4.2. *There is a polynomial time kernelization algorithm for the weighted r-SET PACKING problem that, on an instance (S, k), produces an instance (S', k) where S' is a subcollection of S and S' has at most $(rk)^r = O(k^r)$ r-sets, such that if S contains k-packings then S' contains a maximum weighted k-packing in S.*

Proof. First note that S' is a subcollection of S. Therefore, if S contains no k-packings, then it is impossible for S' to contain a k-packing. Therefore, in the following, we can assume that S contains k-packings.

On a collection S of more than $(rk)^r$ r-sets, our kernelization algorithm works by repeatedly applying the following procedure:

```
Reduce($S$)
1.   for $q = 0$ to $r - 1$ do
1.1      for every $r - q$ elements $a_1, \ldots, a_{r-q}$ in Val($S$) do
1.2          if there are more than $(rk)^q$ $r$-sets in $S$ that contain all $a_1, \ldots, a_{r-q}$
1.3          then remove from $S$ the $r$-set $\sigma_t$ that contains all $a_1, \ldots, a_{r-q}$
1.4                   and has the smallest weight, Return;
2.   remove from $S$ the $r$-set $\sigma'_t$ that has the smallest weight; Return.
```

We must show that the procedure **Reduce** works correctly.

Case A. The procedure **Reduce**(S) returns at step 1.4. Then there are an integer q, $0 \leq q \leq r - 1$, and $r - q$ elements $a_1, \ldots, a_{r-q}$ such that S has more than $(rk)^q$ r-sets that contain all $a_1, \ldots, a_{r-q}$. If this is in Case 1 or Case 2.1 as described in the proof of Lemma 4.1, then as explained in the proof, the proper subcollection $S' = S - \{\sigma_t\}$ of S contains a maximum weighted k-packing in S, where σ_t is the r-set that contains all $a_1, \ldots, a_{r-q}$ and has the smallest weight. On the other hand, Case 2.2 in the proof cannot hold true here: Case 2.2 would imply that there are $r - (q - 1)$ elements $a_1, \ldots, a_{r-q}, b$ such that the collection S has more than $(rk)^{q-1}$ r-sets that contain all $a_1, \ldots, a_{r-q}, b$ (see the discussion of Case 2.2 in the proof of Lemma 4.1). If this were the case, then the procedure would have returned during the $(q - 1)$st execution of the loop of steps 1.1–1.4. In conclusion, if there are an integer q, $0 \leq q \leq r-1$, and $r-q$ elements $a_1, \ldots, a_{r-q}$ such that S has more than $(rk)^q$ r-sets that contain all $a_1, \ldots, a_{r-q}$, then the procedure **Reduce**(S) will produce the proper subcollection $S' = S - \{\sigma_t\}$ of S that contains a maximum weighted k-packing in S.

Case B. The procedure **Reduce**(S) returns at step 2. Let P^* be a maximum weighted k-packing in S. Let $S = \{\sigma'_1, \ldots, \sigma'_h, \sigma'_{h+1}, \ldots, \sigma'_t\}$, where the r-sets have been sorted by their weights in non-increasing order, $h = (rk)^r$, and $t > h$. Let $S_h = \{\sigma'_1, \ldots, \sigma'_h\}$. If all r-sets in P^* are in S_h, then the proper subcollection $S' = S - \{\sigma'_t\}$ contains the maximum weighted k-packing P^*. On the other hand, suppose that there are r-sets in P^* that are not in S_h. Let σ'_j be the r-set in P^* that is not in S_h and has the largest index j in S (in particular, $j > h$). We divide this case into two subcases:

Subcase B.1. There is an r-set σ'_i in S_h, $i \leq h$, that contains no elements in Val(P^*). Then the proper subcollection $S' = S - \{\sigma'_t\}$ will contain $(P^* - \sigma'_j) \cup \{\sigma'_i\}$, which is also a maximum weighted k-packing in S.

Subcase B.2. Every r-set in S_h contains at least one element in Val(P^*). Then S_h plus σ'_j gives a collection of more than $(rk)^r$ r-sets in S in which each r-set contains at least one element in Val(P^*). Since Val(P^*) has exactly rk elements, we derive that there is an element a_1 in Val(P^*) such that $S_h \cup \{\sigma'_j\}$ (thus S) has more than $(rk)^{r-1}$ r-sets that contain a_1. That is, for $q = r-1$, there is $r-q = 1$ element a_1 such that S has more than $(rk)^q$ r-sets that contain a_1. But this is a contradiction: the existence of the element a_1 would have made the procedure **Reduce**(S) to return at step 1.4 during its $(r - 1)$st execution of the loop 1.1–1.4, instead of at step 2. Therefore, Subcase B.2 is impossible.

This completes the proof that if the procedure **Reduce**(S) returns at step 2, then the proper subcollection $S' = S - \{\sigma'_t\}$ must contain a maximum weighted k-packing in S.

Summarizing the discussion in Cases A and B verifies that if the collection S has more than $(rk)^r$ r-sets, then the procedure **Reduce**(S) always returns a proper subcollection S' of S that contains a maximum weighted k-packing in S.

It is easy to derive that the running time of the procedure **Reduce**(S) is bounded by

$$O\left(m \sum_{q=0}^{r} \binom{n}{r-q}\right) = O(mn^r),$$

where m is the number of r-sets in S and $n = |\text{Val}(S)|$.

Therefore, in time $O(mn^r)$, the procedure **Reduce**(S), on a collection S of more than $(rk)^r$ r-sets, produces a proper subcollection S' of S that contains a maximum weighted k-packing in S. If we repeatedly apply the procedure **Reduce** to the resulting collection as long as it contains more than $(rk)^r$ r-sets, for at most $m - (rk)^r$ times, we end up with a subcollection S' of the original collection S such that S' contains at most $(rk)^r$ r-sets and S' contains a maximum weighted k-packing in S. The instance (S', k) then is the desired instance in the theorem, and it can be constructed from the original instance in time $O(m^2 n^r)$. □

Since r-SET PACKING is a natural generalization of rD-MATCHING. Theorem 4.2 also gives a kernelization algorithm for the weighted rD-MATCHING problem.

Finally, we remark that polynomial time kernelization algorithms for the unweighted rD-MATCHING and r-SET PACKING problems have been reported in the literature [5]. On the other hand, Theorem 4.2 seems to be the first kernelization result for the weighted versions of the problems.

5. Conclusions

Matching and packing have been an important class of NP-hard problems. Parameterized algorithms for matching and packing problems have been an active research direction in the past decade. In this paper, we offered new observations that enabled us to take advantage of the recent techniques based on *divide-and-conquer* in parameterized algorithms [2], and develop improved deterministic algorithms for the weighted rD-MATCHING and r-SET PACKING problems for any fixed constant r.

An obvious open problem is whether or not the algorithms can be further improved. In particular, is it possible to develop deterministic algorithms of running time $O^*(2^{rk})$ for the rD-MATCHING and r-SET PACKING problems, even just for the unweighted versions? The approach based on (n, k)-universal set, which has been adopted in the current paper, seems to have its limit to reach this goal. On the other hand, there have been recently developed randomized algorithms of running time $O^*(2^{3k})$ for the unweighted 3D-MATCHING and 3-SET PACKING problems [10], although it is unknown whether these algorithms can be derandomized without significantly increasing the running time and whether these algorithms can be extended to solve the weighted versions of the problems and to solve the problems for a general integer $r > 3$ [10,17].

Theorem 4.2 shows a kernelization upper bound $O(k^r)$ for the weighted rD-MATCHING and the weighted r-SET PACKING problems. This result may lead to further interesting research. In particular, Dell and van Melkebeek [3] have recently been able to establish non-linear lower bounds for kernel size for a variety of parameterized problems. It will be interesting to study nontrivial lower bounds for kernel size for rD-MATCHING and r-SET PACKING, and see how closely the upper bounds presented in the current paper match the lower bounds.

Acknowledgement

The authors thank an anonymous referee for his pointer to the recent work of Dell and van Melkebeek. Moreover, his comments and suggestions have improved the presentation of this paper.

References

[1] J. Chen, D. Friesen, W. Jia, I. Kanj, Using nondeterminism to design effcient deterministic algorithms, Algorithmica 40 (2004) 83–97.
[2] J. Chen, J. Kneis, S. Lu, D. Mölle, S. Richter, P. Rossmanith, S. Sze, F. Zhang, Randomized divide-and-conquer: improved path, matching, and packing algorithms, SIAM Journal of Computing 38 (2009) 2526–2547.
[3] H. Dell, D. van Melkebeek, Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses, in: Proc. 42nd ACM Symp. Theory of Computing, STOC 2010, 2010, pp. 251–260.
[4] R. Downey, M. Fellows, Parameterized Complexity, Springer, New York, 1999.
[5] M. Fellows, C. Knauer, N. Nishimura, P. Ragde, F. Rosamond, U. Stege, D. Thilikos, S. Whitesides, Faster fixed-parameter tractable algorithms for matching and packing problems, Algorithmica 52 (2008) 167–176.
[6] M. Fredman, J. Komlos, E. Szemeredi, Storing a sparse table with $O(1)$ worst case access time, Journal of the ACM 31 (1984) 538–544.
[7] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.
[8] W. Jia, C. Zhang, J. Chen, An effcient parameterized algorithm for m-set packing, Journal of Algorithms 50 (2004) 106–117.
[9] I. Koutis, A faster parameterized algorithm for set packing, Information Processing Letters 94 (2005) 7–9.
[10] I. Koutis, Faster algebraic algorithms for path and packing problems, in: Proc. 35th International Colloquium on Automata, Languages and Programming, ICALP 2008, in: Lecture Notes in Computer Science, 5125, 2008, pp. 575–586.
[11] Y. Liu, J. Chen, J. Wang, On effcient FPT algorithms for weighted matching and packing problems, in: Proc. 4th Ann. Conference on Theory and Applications of Models of Computation, TAMC 2007, in: Lecture Notes in Computer Science, vol. 4484, 2007, pp. 575–586.
[12] Y. Liu, S. Lu, J. Chen, S.H Sze, Greedy localization and color-coding: improved matching and packing algorithms, in: Proc. 2nd International Workshop on Parameterized and Exact Computation, IWPEC 2006, in: Lecture Notes in Computer Science, vol. 4169, 2006, pp. 84–95.
[13] M. Naor, L. Schulman, A. Srinivasan, Splitters and near-optimal derandomization, in: Proc. 39th Annual Symposium on Foundatins of Computer Science, FOCS 1995, 1995, pp. 182–190.
[14] V. Shoup, A Computational Introduction to Number Theory and Algebra, 2nd ed., Cambridge Univesity Press, New York, 2008.
[15] J. Wang, Q. Feng, Improved parameterized algorithms for weighted 3-set packing, in: Proc. 14th Annual International Computing and Combinatorics Conference, COCOON 2008, in: Lecture Notes in Computer Science, vol. 5092, 2008, pp. 130–139.
[16] J. Wang, Q. Feng, An $O^*(3.52^{3k})$ parameterized algorithm for 3-set packing, in: Proc. 5th Annual Conference on Theory and Applications of Models of Computation, TAMC 2008, in: Lecture Notes in Computer Science, vol. 4978, 2008, pp. 82–93.
[17] R. Williams, Finding paths of length k in $O^*(2^k)$ time, Information Processing Letters 109 (2009) 315–318.