Algorithmica (2008) 52: 167–176
DOI 10.1007/s00453-007-9146-y

Faster Fixed-Parameter Tractable Algorithms for Matching and Packing Problems

M.R. Fellows · C. Knauer · N. Nishimura · P. Ragde · F. Rosamond · U. Stege · D.M. Thilikos · S. Whitesides

Received: 31 October 2006 / Accepted: 26 November 2007 / Published online: 7 December 2007

Abstract We obtain faster algorithms for problems such as r-dimensional matching and r-set packing when the size k of the solution is considered a parameter. We first establish a general framework for finding and exploiting small problem kernels (of size polynomial in k). This technique lets us combine Alon, Yuster and Zwick's color-coding technique with dynamic programming to obtain faster fixed-parameter algo-

Research initiated at the International Workshop on Fixed Parameter Tractability in Computational Geometry and Games, Bellairs Research Institute of McGill University, Holetown, Barbados, Feb. 7–13, 2004, organized by S. Whitesides.
D.M. Thilikos supported by the EU within the 6th Framework Programme under contract 001907 (DELIS) and by the Spanish CICYT project TIC-2002-04498-C05-03 (TRACER).

M.R. Fellows · F. Rosamond
School of Electrical Engineering and Computer Science, University of Newcastle, Newcastle, Australia

C. Knauer
Institute of Computer Science, Freie Universität Berlin, Berlin, Germany

N. Nishimura · P. Ragde (✉)
School of Computer Science, University of Waterloo, Waterloo, Canada
e-mail: plragde@uwaterloo.ca

N. Nishimura
e-mail: nishi@uwaterloo.ca

U. Stege
Department of Computer Science, University of Victoria, Victoria, Canada

D.M. Thilikos
Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain

S. Whitesides
School of Computer Science, McGill University, Montreal, Canada

rithms for these problems. Our algorithms run in time $O(n+2^{O(k)})$, an improvement over previous algorithms for some of these problems running in time $O(n+k^{O(k)})$. The flexibility of our approach allows tuning of algorithms to obtain smaller constants in the exponent.

Keywords Parameterized complexity · Fixed parameter tractable · Graph matching · Set packing · Color coding

1 Introduction

In this paper we demonstrate a general method for solving parameterized packing and matching problems by first finding a problem kernel, and then showing how color-coding (the use of nice families of hash functions to color a substructure with distinct colors) and dynamic programming on subsets of colors can be used to create fixed-parameter algorithms. The problems we consider include parameterized versions of r-dimensional matching (a generalization of 3-dimensional matching, from Karp's original list of NP-complete problems [11]) and r-set packing (on Karp's list in its unrestricted form, and $W[1]$-complete [2] in its unrestricted parameterized form, hence unlikely to be fixed-parameter tractable).

Previous efforts in finding fixed-parameter algorithms for these problems (asking if a given input has a solution of size at most k) have resulted in running times involving a factor of $k^{O(k)}$, namely $O((5.7k)^k \cdot n)$ for 3-dimensional matching [3] and 3-set packing [10]. Earlier work did not use the technique of kernelization (finding a subproblem of size $f(k)$ within which any solution of size k must lie) except for work on the problem of packing triangles in a graph [7]; that result uses the technique of crown decomposition [5, 6], a technique that is not used in this paper. Using our techniques, we improve all these results by replacing the $k^{O(k)}$ factor with a factor of $2^{O(k)}$. Kernelization also allows us to make the dependence on k additive, that is, we can achieve a running time of $O(n+2^{O(k)})$ (the hidden constant in the exponent is linearly dependent on r). We note also that independently, Koutis [13] obtained a similar improvement, though with worse constants in the exponent.

The techniques we introduce make use of the fact that our goal is to obtain at least k disjoint objects, each represented as a tuple of values. We first form a maximal set of disjoint objects and then use this set as a way to bound the number of values outside the set, forming the kernel which is used in the color-coding dynamic programming. We are able to reduce the size of the constants inherent in the choice of hash functions in the original formulation of color-coding [1] by applying the coloring to our kernels only, and using smaller families of hash functions with weaker properties. Although both color-coding and dynamic programming across subsets appear in other work [5, 15, 17], it is our kernelization technique that permits the improvements. Recent work [4, 12, 14] has produced further improvements through the use of more sophisticated methods.

After introducing notation common to all the problems in Sect. 2, we first present the kernelization algorithm for r-DIMENSIONAL MATCHING (Sect. 3), followed by the fixed-parameter algorithm in Sect. 4. Next, we consider the modifications necessary to solve the other problems in Sect. 5. The paper concludes with a discussion of tuning the hash functions (Sect. 6).

2 Definitions

Each of the problems considered in this paper requires the determination of k or more disjoint structures within the given input: in r-dimensional matching, we find k points so that no coordinates are repeated, and in r-set packing, we find k disjoint sets. To unify the approach taken to the problems, we view each structure as an r-tuple; the problems differ in the ways the r-tuples must be disjoint.

To define the r-tuples, we consider a collection $A_1, \ldots, A_r$ of pair-wise disjoint sets and define $\mathcal{A}$ to be $A_1 \times \cdots \times A_r$. We call the elements of each A_i *values*; given an r-tuple $T \in \mathcal{A}$, we denote as $\mathsf{val}(T)$ the set of values of T, and for any $\mathcal{S} \subseteq \mathcal{A}$, we define $\mathsf{val}(\mathcal{S}) = \bigcup_{T \in \mathcal{S}} \mathsf{val}(T)$. Depending on the problem, we will either wish to ensure that tuples chosen have disjoint sets of values or, for r-dimensional matching, that tuples "disagree" at a particular coordinate.

We introduce further notation to handle the discussion of r-tuples. Given a tuple $T \in \mathcal{A}$, the *ith coordinate of* T, denoted $T(i)$, is the value of T from set A_i. Two r-tuples T and T' are *linked* if they agree in some coordinate i, that is, when there exists some i in $\{1, \ldots, r\}$ such that $T(i) = T'(i)$. We denote this fact as $T \sim T'$. If $T \sim T'$ does not hold then we say that T and T' are *independent*, denoted $T \not\sim T'$.

We can now define our first problem, r-DIMENSIONAL MATCHING, in which the input is a set of r-tuples and the goal is to find at least k independent tuples.

r-DIMENSIONAL MATCHING
Instance: A set $\mathcal{S} \subseteq \mathcal{A} = A_1 \times \cdots \times A_r$ for some collection $A_1, \ldots, A_r$ of pair-wise disjoint sets.
Parameter: A non-negative integer k.
Question: Is there a matching $\mathcal{P}$ for $\mathcal{S}$ of size at least k, that is, is there a subset $\mathcal{P}$ of $\mathcal{S}$ where for any $T, T' \in \mathcal{P}$, $T \not\sim T'$ and $|\mathcal{P}| \geq k$?

In forming a kernel, we will be able to reduce the number of tuples that contain a particular value or set of values. To describe a set of tuples in which some values are specified and others may range freely, we will need an augmented version A_i^* of each A_i so that $A_i^* = A_i \cup \{*\}$, $i \in \{1 \ldots, r\}$; the stars will be used to denote positions at which values are unrestricted. We will refer to special tuples, *patterns*, drawn from the set $\mathcal{A}^* = A_1^* \times \cdots \times A_r^*$. Associated with each pattern will be a corresponding set of tuples. More formally, given $R \in \mathcal{A}^*$, we set $\mathcal{A}[R] = A_1' \times \cdots \times A_r'$ where

$$A_i' = \begin{cases} A_i & \text{if } R(i) = *, \\ \{R(i)\} & \text{otherwise.} \end{cases}$$

As we will typically wish to consider the subset of $\mathcal{S}$ extracted using the pattern, we further define, for any $\mathcal{S} \subseteq \mathcal{A}$, the set $\mathcal{S}[R] = \mathcal{A}[R] \cap \mathcal{S}$. We finally define $\mathsf{free}(R)$ as the number of $*$'s in R, namely the number of unrestricted positions in the pattern; $\mathsf{free}(R)$ will be called the *freedom of* R.

3 A Kernel for r-DIMENSIONAL MATCHING

Recall that a kernel is a subproblem of size depending only on k within which any solution of size k must lie. The idea behind our kernelization technique is that if a

large number of tuples match a particular pattern, we can remove some of them (this is known as a *reduction rule*). For each pattern R, the threshold size for the set of retained tuples is a function $f(\mathsf{free}(R), k)$, which we define later.

The following gives a general family of reduction rules for the r-DIMENSIONAL MATCHING problem with input $\mathcal{S}_I$ and parameter k. For each application of the function Reduce, if the size of the subset $\mathcal{S}_I[R]$ of $\mathcal{S}$ extracted using the pattern R is greater than the threshold, then $|\mathcal{S}|$ is reduced by removing enough elements of $\mathcal{S}_I[R]$ to match the size of the function f.

Function $\mathsf{Reduce}(\mathcal{S}_I, R)$ where $R \in \mathcal{A}^*$ and $1 \leq \mathsf{free}(R) \leq r-1$.
Input: A set $\mathcal{S}_I \subseteq \mathcal{A}$ and a pattern $R \in \mathcal{A}^*$.
Output: A set $\mathcal{S}_O \subseteq \mathcal{S}_I$.

$\mathcal{S}_O \leftarrow \mathcal{S}_I$.
If $|\mathcal{S}_I[R]| > f(\mathsf{free}(R), k)$
then remove all but $f(\mathsf{free}(R), k)$ tuples of $\mathcal{S}_I[R]$ from $\mathcal{S}_O$.
output $\mathcal{S}_O$.

To form the kernel, we apply the function Reduce on all patterns $R \in \mathcal{A}^*$ in order of increasing freedom, as in the following routine.

Function Fully-Reduce($\mathcal{S}_I$)
Input: A set $\mathcal{S}_I \subseteq \mathcal{A}$.
Output: A set $\mathcal{S}_O \subseteq \mathcal{S}_I$.

$\mathcal{S}_O \leftarrow \mathcal{S}_I$.
For $i = 1, \ldots, r-1$ **do**
  for all $R \in \mathcal{A}^*$ where $\mathsf{free}(R) = i$, $\mathcal{S}_O \leftarrow \mathsf{Reduce}(\mathcal{S}_O, R)$.
output $\mathcal{S}_O$.

The next three lemmas prove that the above function computes a kernel; the two lemmas after that show how a slight modification can be used to bound the size of that kernel.

We say that a set $\mathcal{S} \subseteq \mathcal{A}$ is (ℓ, k)-*reduced* if for any $R \in \mathcal{A}^*$ such that $\mathsf{free}(R) = \ell$, $\mathsf{Reduce}(\mathcal{S}, R) = \mathcal{S}$ (where $1 \leq \ell \leq r-1$). For notational convenience we define any set $\mathcal{S} \subseteq \mathcal{A}$ to be 0-reduced; we can assume the input has no repeated tuples. As a direct consequence of the definition, if a set $\mathcal{S}$ is (ℓ, k)-reduced, then for each pattern R such that $\mathsf{free}(R) = \ell$, $|\mathcal{S}[R]| \leq f(\ell, k)$.

We now show that our reduction function does not change a yes-instance of r-DIMENSIONAL MATCHING into a no-instance, nor vice versa. Given a set $\mathcal{S} \subseteq \mathcal{A}$ and a non-negative integer k, we denote as $\mu(\mathcal{S}, k)$ the set of all matchings for $\mathcal{S}$ of size at least k. If $\mu(\mathcal{S}, k)$ is non-empty, $\mathcal{S}$ is a yes-instance. The following lemma shows that upon applying the reduction rule to an $(\ell-1, k)$-reduced set using a pattern R of freedom ℓ, yes-instances will remain yes-instances. The proof requires the precise definition of $f(\ell, k)$, which is that $f(0, k) = 1$ and $f(\ell, k) = \ell \cdot (k-1) \cdot f(\ell-1, k) + 1$ for all $\ell > 0$. It is easy to see that $f(\ell, k) \leq \ell! k^\ell$.

Lemma 1 *For any* $\ell, 1 \leq \ell \leq r-1$, *the following holds*: *If* $\mathcal{S} \subseteq \mathcal{A}$, $\mathcal{S}$ *is* $(\ell-1,k)$*-reduced, and* $\mathcal{S}' = \mathsf{Reduce}(\mathcal{S}, R)$ *for some* R *such that* $\mathsf{free}(R) = \ell$, *then* $\mu(\mathcal{S},k) \neq \emptyset$ *if and only if* $\mu(\mathcal{S}',k) \neq \emptyset$.

Proof Since $\mathcal{S}' \subseteq \mathcal{S}$, $\mu(\mathcal{S}',k) \neq \emptyset$ implies $\mu(\mathcal{S},k) \neq \emptyset$. Supposing now that $\mu(\mathcal{S},k) \neq \emptyset$, we choose $\mathcal{P} \in \mu(\mathcal{S},k)$ where $|\mathcal{P}| = k$ and prove that $\mu(\mathcal{S}',k) \neq \emptyset$. In essence, we need to show that either $\mathcal{P} \in \mu(\mathcal{S}',k)$ or that we can form a matching $\mathcal{P}' \in \mu(\mathcal{S}',k)$ using some of the tuples in $\mathcal{P}$.

Denote by $\hat{\mathcal{S}}$ the set $\mathcal{S}'[R]$, that is, the tuples retained when Reduce is applied to $\mathcal{S}$ using pattern R to form $\mathcal{S}'$. Clearly if either $\mathcal{P}$ contained no tuple in $\mathcal{S}[R]$ or the single tuple $T \in \mathcal{P} \cap \mathcal{S}[R]$ was selected to be in $\hat{\mathcal{S}}$, then $\mathcal{P} \in \mu(\mathcal{S}',k)$, completing the proof in these two cases. More formally, we can describe these situations using a partition of the set of indices into $I_R = \{i \mid R(i) \neq *\}$, the indices that are not stars, and $I_R^* = \{i \mid R(i) = *\}$, the indices that are stars. In the first case, no tuple $T \in \mathcal{P}$ exists where for all $i \in I_R$ $T(i) = R(i)$, and hence $\mathcal{P} \cap \mathcal{S}[R] = \emptyset$. This implies that $\mathcal{P} \subseteq \mathcal{S}'$ and hence the proof is complete as $\mathcal{P} \in \mu(\mathcal{S}',k)$. In the second case, for some $T \in \mathcal{P}$, for all $i \in I_R$ we have $T(i) = R(i)$, and so if $T \in \hat{\mathcal{S}}$ then $\mathcal{P} \subseteq \mathcal{S}'$, as needed.

In the case where $T \in \mathcal{P}$ and $T \notin \hat{\mathcal{S}}$, we show that there is a tuple $T' \in \hat{\mathcal{S}}$ that can replace T to form a new matching, formalized in the claim below.

Claim *There exists a* $T' \in \hat{\mathcal{S}}$ *that is independent from each* $\tilde{T} \in \mathcal{P} - \{T\}$.

Proof We first show that for any $i \in I_R^*$ and any $\tilde{T} \in \mathcal{P} - \{T\}$ there exist at most $f(\ell-1,k)$ r-tuples in $\hat{\mathcal{S}}$ that agree with $\tilde{T}$ at position i. The set of r-tuples in $\mathcal{S}[R]$ that agree with $\tilde{T}$ at position i is exactly the set of tuples in $\mathcal{S}[R']$ where R' is obtained from R by replacing the $*$ in position i by $\tilde{T}(i)$. We use the size of this set to bound the size of the set of tuples in $\hat{\mathcal{S}}$ that agree with $\tilde{T}$ at position i. As $\mathcal{S}$ is $(\ell-1,k)$-reduced and $\mathsf{free}(R') = \ell-1$, we know that $|\mathcal{S}[R']| \leq f(\ell-1,k)$. Since in the special case where $\ell = 1$, we directly have $|\mathcal{S}[R']| \leq 1 = f(0,k)$, the bound of $f(\ell-1,k)$ holds for any $\tilde{T}$ and any i.

To complete the proof of the claim, we determine the number of elements of $\hat{\mathcal{S}}$ that can be linked with any $\tilde{T}$ and show that at least one member of $\hat{\mathcal{S}}$ is not linked with any in the set $\mathcal{P} - \{T\}$. If we fix a tuple $(\tilde{T})$ and a free position $i \in I_R^*$, the statement proved in the paragraph above says that there are at most $f(\ell-1,k)$ r-tuples in $\hat{\mathcal{S}}$ linked with it. But there are ℓ choices of free position for each tuple $\tilde{T}$, and $|\mathcal{P} - \{T\}| = k-1$ choices of tuples $\tilde{T}$. Thus at most $\ell \cdot (k-1) \cdot f(\ell-1,k) = f(\ell,k) - 1$ elements of $\hat{\mathcal{S}}$ will be linked with elements of $\mathcal{P} - \{T\}$. Since $|\hat{\mathcal{S}}| = f(\ell,k)$, this implies the existence of some $T' \in \hat{\mathcal{S}}$ with the required property. □

The claim above implies that $\mathcal{P}' = (\mathcal{P} - \{T\}) \cup \{T'\}$ is a matching of $\mathcal{S}$ of size k. As $T' \in \hat{\mathcal{S}}$, $\mathcal{P}'$ is also a matching of $\mathcal{S}'$, and thus $\mu(\mathcal{S}',k) \neq \emptyset$. □

Lemma 2 *If* $\mathcal{S} \subseteq \mathcal{A}$ *and* $\mathcal{S}'$ *is the output of routine* Fully-Reduce$(\mathcal{S})$, *then* (a) $\mathcal{S}'$ *is an* $(r-1,k)$*-reduced set and* (b) $\mu(\mathcal{S},k) \neq \emptyset$ *if and only if* $\mu(\mathcal{S}',k) \neq \emptyset$.

Proof The proof of part (a) follows from induction on i, the number of iterations of the loop, and two simple observations: for any $R \in \mathcal{A}^*$ and any $(\mathsf{free}(R)-1,k)$-reduced set $\mathcal{S} \subseteq \mathcal{A}$, $\mathsf{Reduce}(\mathcal{S}, R)$ is also a $(\mathsf{free}(R)-1,k)$-reduced set; and after the application of Reduce for every R such that $\mathsf{free}(R) = \ell$, the resulting set is (ℓ,k)-reduced. Since at the outset S_I is trivially 0-reduced, at the end of iteration $r-1$ claim (a) will hold.

Part (b) can also be proved by induction on the number of iterations of the loop by showing that for any iteration $\mu(\mathcal{S}_b,k) \neq \emptyset$ if and only if $\mu(\mathcal{S}_a,k) \neq \emptyset$, where $\mathcal{S}_b$ is equal to $\mathcal{S}_O$ before the iteration and $\mathcal{S}_a$ is equal to $\mathcal{S}_O$ after the iteration. In particular, we apply Lemma 1 repeatedly within each iteration (once for each pattern R used). By repeating this for $\ell = 1,\ldots,r-1$ we finally transform any set $\mathcal{S}$ into a $(r-1,k)$-reduced set $\mathcal{S}'$ such that $\mu(\mathcal{S},k) \neq \emptyset$ if and only if $\mu(\mathcal{S}',k) \neq \emptyset$. □

We now use a maximal matching in conjunction with the function Fully-Reduce defined above to bound the size of the kernel. The following lemma is a direct consequence of the definition of an $(r-1,k)$-reduced set.

Lemma 3 *For any $(r-1,k)$-reduced set $\mathcal{S} \subseteq \mathcal{A}$, any value $x \in \mathsf{val}(\mathcal{A})$ is contained in at most $f(r-1,k)$ r-tuples of $\mathcal{S}$.*

We now observe that if we have a maximal matching in a set $\mathcal{S}$ of r-tuples, we can bound the size of the set as a function of r, the size of the matching, and $f(r-1,k)$, as each r-tuple in the set must be linked with at least one r-tuple in the matching, and the number of such links is bounded by $f(r-1,k)$ per value.

Lemma 4 *If $\mathcal{S} \subseteq \mathcal{A}$ is an $(r-1,k)$-reduced set containing a maximal matching $\mathcal{M}$ of size at most m, then $\mathcal{S}$ contains no more than $r \cdot m \cdot f(r-1,k)$ r-tuples.*

Proof Suppose that $\mathcal{M}$ is a maximal matching of size at most m in $\mathcal{S}$. Clearly, $|\mathsf{val}(\mathcal{M})| \leq r \cdot m$. As $\mathcal{M}$ is maximal, for any r-tuple $T \in \mathcal{S}$, $\mathsf{val}(T) \cap \mathsf{val}(\mathcal{M}) \neq \emptyset$. From Lemma 3 we know that each value in $\mathsf{val}(\mathcal{M})$ appears in at most $f(r-1,k)$ r-tuples of $\mathcal{S}$. Since any r-tuple of $\mathcal{S}$ contains at least one of the $r \cdot m$ values in $\mathsf{val}(\mathcal{M})$, we conclude that $|\mathcal{S}| \leq r \cdot m \cdot f(r-1,k)$. □

Lemma 4 suggests a kernel for the r-DIMENSIONAL MATCHING problem in the function that follows; Lemma 5 shows the correctness of the procedure and size of the kernel.

Function Kernel-Construct($\mathcal{S}$)
Input: A set $\mathcal{S} \subseteq \mathcal{A}$.
Output: A set $\mathcal{K} \subseteq \mathcal{S}$.

$\mathcal{S}' \leftarrow$Fully-Reduce($\mathcal{S}$).
Find a maximal matching $\mathcal{M}$ of $\mathcal{S}'$.
If $|\mathcal{M}| \geq k$ then **output** any subset $\mathcal{K}$ of $\mathcal{M}$ of size k and **stop**;
otherwise **output** $\mathcal{K} \leftarrow \mathcal{S}'$.

Lemma 5 *If $\mathcal{S} \subseteq \mathcal{A}$ and $\mathcal{K}$ is the output of the function* Kernel-Construct(S), *then* (a) $|\mathcal{K}| \in O(r!k^r)$ *and* (b) $\mu(\mathcal{S}, k) \neq \emptyset$ *if and only if* $\mu(\mathcal{K}, k) \neq \emptyset$.

Proof To see that part (a) holds, if $|\mathcal{M}| \leq k-1$, then since $\mathcal{K} = \mathcal{S}'$ is an $(r-1, k)$-reduced set, by Lemma 4 $|\mathcal{K}| \leq r \cdot (k-1) \cdot f(r-1, k) \leq r!k^r$. If instead $|\mathcal{M}| \geq k$, then $|\mathcal{K}| = k \in O(r!k^r)$.

To prove part (b), we first observe that since $\mathcal{K} \subseteq \mathcal{S}$, clearly $\mu(\mathcal{K}, k) \subseteq \mu(\mathcal{S}, k)$, and hence if $\mu(\mathcal{K}, k) \neq \emptyset$, then $\mu(\mathcal{S}, k) \neq \emptyset$. Suppose now that $\mu(\mathcal{S}, k) \neq \emptyset$. In the case $\mathcal{K} = \mathcal{S}'$, by Lemma 2, $\mathcal{S}'$ is $(r-1, k)$-reduced and $\mu(\mathcal{S}', k) \neq \emptyset$. In the case $\mathcal{K} \subseteq \mathcal{M}$ the result $\mu(\mathcal{K}, k) \neq \emptyset$ follows trivially. □

Note that function Kernel-Construct can be computed in time $O(n)$ for fixed r, simply by looking at each tuple in turn and incrementing the counter associated with each of the $2^r - 1$ patterns derived from it, deleting the tuple if any such counter overflows its threshold.

4 An FPT Algorithm for r-Dimensional Matching

While it suffices to restrict attention to the kernel $\mathcal{K}$ when seeking a matching of size k, exhaustive search of $\mathcal{K}$ does not lead to a fast algorithm. To obtain such an algorithm, we combine the color coding technique of Alon et al. [1] with dynamic programming on subsets of colors.

In its original form, the color-coding technique of Alon et al. makes use of a family of hash functions $\mathcal{F} = \{f : U \to X\}$ with the property that for any $S \subseteq U$ with $|S| \leq |X|$, there is an $f \in \mathcal{F}$ that is 1-1 on X. The original idea, in the context of our problem, would be to look for a matching of size k whose values receive distinct colors under some $f \in \mathcal{F}$. In our case, the universe U is the set of values appearing in tuples in the kernel of the problem, which has size $O(k^r)$, and the set of colors X has size rk. The family of hash functions $\mathcal{F}$ used by Alon et al. is of size bounded by $2^{c|X|}$, which is $2^{O(rk)}$ in our case. In Sect. 6 we will consider modifications towards improving this approach.

The conference presentation of this work [8] gave a more complicated algorithm. We thank Alexander Golynski who, on seeing our exposition, pointed out a simplification which we present here. In the simple algorithm, the dynamic programming space has two dimensions: the choice of hash function ϕ, and a set of values Y used by a matching. We define

$$B_\phi(Y) = \begin{cases} 1 & \text{if there exists a matching } \mathcal{P}' \subseteq \mathcal{S} \text{ where } \mathsf{val}(\mathcal{P}') = Y, \\ 0 & \text{otherwise.} \end{cases}$$

The number of dynamic programming problems thus defined is $2^{O(rk)}$, since there are $2^{O(rk)}$ choices for ϕ and 2^{rk} choices for Y. The base case for the dynamic programming recurrence is $B_\phi(\emptyset) = 1$, and the recurrence is:

$$B_\phi(Y) = \begin{cases} 1 & \text{if there exists an } r\text{-tuple } T \text{ where } B_\phi(Y - \mathsf{val}(T)) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

The cost of computing one table entry is $O(k^r)$, since this is the number of tuples in $\mathcal{K}$. Noting that kernel construction takes time $O(n)$, we have the following theorem.

Theorem 1 *The problem* r-DIMENSIONAL MATCHING *can be solved in time* $O(n + 2^{(c+1)rk})$.

5 A Kernel and FPT Algorithm for Set Packing

We can now define other problems addressable by our technique. For r-SET PACKING, the input r-tuples are subsets of elements from a base set A, each of size at most r, and the goal is to find at least k r-tuples, none of which share elements. More formally:

r-SET PACKING
Instance: A collection $\mathcal{C}$ of sets drawn from a set A, each of size of at most r.
Parameter: A non-negative integer k.
Question: Does $\mathcal{C}$ contain at least k mutually disjoint sets, that is, for $\mathcal{S} \subseteq \mathcal{A} = A^r$, is there a subset $\mathcal{P}$ of $\mathcal{S}$ where for any $T, T' \in \mathcal{P}$, $\mathsf{val}(T) \cap \mathsf{val}(T') = \emptyset$ and $|\mathcal{P}| \geq k$?

To obtain a kernel for r-SET PACKING, we can adapt the ideas developed for r-DIMENSIONAL MATCHING. To avoid introducing new notation, we will continue to refer to sets of tuples. As with r-DIMENSIONAL MATCHING, we first apply a reduction rule that limits the number of tuples containing a particular value or set of values, and then use a maximal set of disjoint objects to bound the size of the kernel.

For r-SET PACKING, the tuples in question are drawn from the same base set (A, $V(G)$, or $E(G)$) respectively, resulting in the tuples being sets of size at most r, as position within a tuple is no longer important. But since the A_i's are no longer disjoint, the potential for conflicts increases. As a consequence, we define a new function g for use in the function Reduce: $g(0, k) = 1$ and $g(\ell, k) = \ell \cdot r(k-1) \cdot g(\ell - 1, k) + 1$ for all $\ell > 0$. By replacing f by g in the definition of the function Reduce, we obtain a new function SetReduce and the notion of a set being (ℓ, k)-*set-reduced*, and by replacing Reduce by SetReduce in the function Fully-Reduce, we can obtain a new function Fully-SetReduce. Instead of finding a maximal matching, we find a maximal set of disjoint sets. It then remains to prove analogues of Lemmas 1 through 5; sketches of the changes are given below.

Each of the lemmas can be modified by replacing f by g, Reduce by SetReduce, and Fully-Reduce by Fully-SetReduce, with the analogue of Lemma 5 yielding a kernel of size $O(k^r)$. The need for g instead of f is evident in the proof of the claim in the analogue of Lemma 1. Here we count the number of r-tuples $\hat{T}$ in $\hat{S}$ such that $\mathsf{val}(\hat{T}) \cap \mathsf{val}(\tilde{T}) \neq \emptyset$. Before, "linked" meant agreeing in some coordinate; now it means "sharing a common value". Hence the counting of such r-tuples changes. For a fixed tuple $\tilde{T}$, there are r choices of value to be shared, and at most $g(\ell - 1, k)$ tuples that can share that value. The number of choices for $\tilde{T}$ is the number of tuples in $\mathcal{P} - \{\mathcal{T}\}$ (that is, $k-1$) multiplied by $g(\ell - 1, k)$. In total this gives us $r \cdot (k-1) \cdot g(\ell - 1, k) = g(\ell, k) - 1$ elements of $\hat{S}$ with nonempty intersection with values in elements of $\mathcal{P} - \{\mathcal{T}\}$, as needed to show that there is a T' with the required property. We then have the following lemma, analogous to Lemma 5:

Lemma 6 *For the problem r-Set Packing, if $\mathcal{S} \subseteq \mathcal{A}$ and $\mathcal{K}$ is the output of the function* Kernel-Construct(S), *then* (a) $|\mathcal{K}| \in O(r^r k^r)$ *and* (b) $\mathcal{S}$ *has a set of at least k disjoint sets if and only if $\mathcal{K}$ has a set of at least k such objects.*

To obtain an algorithms for r-Set Packing, we adapt the ideas developed for r-Dimensional Matching; as in that case we form a kernel, find a maximal set of disjoint sets, and then use dynamic programming and color-coding to find a solution of size k, if one exists.

The analysis of the algorithms depends on Lemma 6, resulting in the theorem below.

Theorem 2 *The problem r-Set Packing can be solved in time $O(n + 2^{O((c+1)rk)})$.*

6 Choosing Hash Functions

To construct the family of hash functions $\mathcal{F} = \{f : U \to X\}$ with the property that for any subset S of U with $|S| = |X|$, there is an $f \in \mathcal{F}$ that is 1-1 on S, Alon et al. used results from the theory of perfect hash functions together with derandomization of small sample spaces that support almost ℓ-wise independent random variables. They were able to construct a family $\mathcal{F}$ with $|\mathcal{F}| = 2^{O(|X|)} \log |U|$, and this is what we used in Sect. 4. However, they made no attempt to optimize the constant hidden in the O-notation, since they were assuming $|X|$ fixed and $|U|$ equal to the size of the input.

In our case (and in any practical implementation of the algorithms of Alon et al.), it would be preferable to lower the constant in the exponent as much as possible. We have one advantage: kernelization means that $|U|$ in our case is $O(k^r)$, not $O(n)$, and so we are less concerned about dependence on $|U|$ (note that $|S| = (r-1)k$ in our case). Two other optimizations are applicable both to our situation and that of Alon et al. First, we can allow more colors, i.e. $|X| = \alpha(r-1)k$ for some constant α. This will increase the number of dynamic programming problems, since the number of choices for C (in determining the number of $B_{\phi,W}(Z, C)$) grows from $2^{(r-1)k}$ to $\binom{\alpha(r-1)k}{(r-1)k} \leq (e\alpha)^{(r-1)k}$. But this may be offset by the reduction in the size of the family of hash functions, since allowing more colors makes it easier to find a perfect hash function. Second, for some of the work on the theory of perfect hash functions used by Alon et al., it is important that the hash functions be computable in $O(1)$ time, whereas in our application, we can allow time polynomial in k.

As an example of applying such optimization, we can make use of the work of Slot and van Emde Boas [16]. They give a scheme based on the pioneering work of Fredman, Komlós, and Szemerédi [9] that in our case results in a family $\mathcal{F}$ of size $2^{4|S|+5\log|S|+3}|U|$, where $|X| = 6|S|$, and it takes $O(k)$ time to compute the value of a hash function. We will not explore this line of improvement further beyond underlining that there is a tradeoff between the power of the family of hash functions we use and the size, and we have some latitude in choosing families with weaker properties.

Another possibility, also discussed by Alon et al., is to replace the deterministic search for a hash function (in a family where the explicit construction is complicated)

by a random choice of coloring. A random $2|S|$-coloring of U is likely to be 1-1 on a subset S with failure probability that is exponentially small in $|S|$. However, this means that a "no" answer has a small probability of error, since it could be due to the failure to find a perfect hash function.

Very recently, Chen et al. report improved bounds on the size of the family $\mathcal{F}$ of hash functions [4]. They show that there is an algorithm that, in $O(6.4^{|X|}|U|)$ steps, can construct a family $\mathcal{F}$ of hash functions with the properties we require, where $|\mathcal{F}| = O(6.4^{|X|}|U|)$. Their paper also improves the upper bound to $O(6.1^{|X|}|U|)$, and discusses consequent improvements on the running times of our algorithms.

References

1. Alon, N., Yuster, R., Zwick, U.: Color-coding. J. Assoc. Comput. Mach. **42**(4), 844–856 (1995)
2. Ausiello, G., D'Atri, A., Protasi, M.: Structure preserving reductions among context optimization problems. J. Comput. Syst. Sci. **21**, 136–153 (1980)
3. Chen, J., Friesen, D.K., Jia, W., Kanj, I.: Using nondeterminism to design deterministic algorithms. In: Proceedings of the 21st Conference on Foundations of Software Technology and Theoretical Computer Science (FST TCS). Lecture Notes in Computer Science, vol. 2245, pp. 120–131. Springer, New York (2001)
4. Chen, J., Lu, S., Sze, S.-H., Zhang, F.: Improved algorithms for path, matching, and packing problems. In: Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 298–307 (2007)
5. Chor, B., Fellows, M.R., Juedes, D.: Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. In: Proceedings of the 30th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2004), pp. 257–269 (2004)
6. Fellows, M.R.: Blow-ups, win/win's, and crown rules: some new directions in FPT. In: Proceedings of the 29th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 03). Lecture Notes in Computer Science, vol. 2880, pp. 1–12. Springer, New York (2003)
7. Fellows, M.R., Heggernes, P., Rosamond, F.A., Sloper, C., Telle, J.A.: Exact algorithms for finding k disjoint triangles in an arbitrary graph. In: Proceedings of the 30th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2004), pp. 257–269 (2004)
8. Fellows, M.R., Knauer, C., Nishimura, N., Ragde, P., Rosamond, F., Stege, U., Thilikos, D.M., Whitesides, S.: Faster fixed-parameter tractable algorithms for matching and packing problems. In: Proceedings of the 12th Annual European Symposium on Algorithms (ESA), pp. 311–322 (2004)
9. Fredman, M.L., Komlós, J., Szemerédi, E.: Storing a sparse table with $o(1)$ worst-case access time. In: Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science, pp. 165–169 (1982)
10. Jia, W., Zhang, C., Chen, J.: An efficient parameterized algorithm for m-set packing. J. Algorithms **50**(1), 106–117 (2004)
11. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum, New York (1972)
12. Kneis, J., Moelle, D., Richter, S., Rossmanith, P.: Divide-and-color. In: Proceedings of the 32nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG), pp. 58–67 (2006)
13. Koutis, I.: A faster parameterized algorithm for set packing. Inf. Process. Lett. **94**, 7–9 (2005)
14. Liu, Y., Lu, S., Chen, J., Sze, S.: Greedy localization and color-coding: improved matching and packing algorithms. In: Proceedings of the 2nd International Workshop on Parameterized and Exact Computation. Lecture Notes in Computer Science, vol. 4162, pp. 84–95. Springer, New York (2006)
15. Marx, D.: Parameterized complexity of constraint satisfaction problems. In: Proceedings of the 19th Annual IEEE Conference on Computational Complexity, 2004
16. Slot, C.F., van Emde Boas, P.: On tape versus core; an application of space efficient perfect hash functions to the invariance of space. Elektron. Inf. Kybern. **21**(4/5), 246–253 (1985)
17. Woeginger, G.J.: Exact algorithms for NP-hard problems, a survey. In: Combinatorial Optimization—Eureka, You Shrink!. Lecture Notes on Computer Science, vol. 2570, pp. 185–207. Springer, New York (2003)