Faster Fixed-Parameter Tractable Algorithms for Matching and Packing Problems*

Michael R. Fellows[1], C. Knauer[2], N. Nishimura[3,**], P. Ragde[3], F. Rosamond[1], U. Stege[4], Dimitrios M. Thilikos[5], and S. Whitesides[6]

[1] School of Electrical Engineering and Computer Science, University of Newcastle, Australia
[2] Institute of Computer Science, Freie Universität Berlin, Germany
[3] School of Computer Science, University of Waterloo, Canada nishi@uwaterloo.ca
[4] Department of Computer Science, University of Victoria, Canada
[5] Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Spain.¶
[6] School of Computer Science, McGill University, Canada

Abstract. We obtain faster algorithms for problems such as r-dimensional matching, r-set packing, graph packing, and graph edge 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). Previously such a kernel was known only for triangle packing. This technique lets us combine, in a new and sophisticated way, Alon, Yuster and Zwick's color-coding technique with dynamic programming on the structure of the kernel to obtain faster fixed-parameter algorithms 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.

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 [Kar72]), r-set

* 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.

** Contact author.

¶ 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).

S. Albers and T. Radzik (Eds.): ESA 2004, LNCS 3221, pp. 311–322, 2004.

packing (on Karp's list in its unrestricted form, and $W[1]$-complete [ADP80] in its unrestricted parameterized form, hence unlikely to be fixed-parameter tractable), graph packing (a generalization of subgraph isomorphism known to be NP-complete [Coo71]), and graph edge packing (shown to be NP-complete by Holyer [Hol81]). Further generalizations are considered at the end of the paper.

Previous efforts in finding fixed-parameter algorithms for these problems (asking if a given input has a solution of size at least k) have resulted in running times involving a factor of $k^{O(k)}$, namely $O((5.7k)^k \cdot n)$ for 3-dimensional matching [CFJK01] and 3-set packing [JZC04]. 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 [FHR+04]; that result uses the technique of crown decomposition [Fel03,CFJ04], a technique that is not used in this paper (concurrent with this paper are further uses of this technique for packing triangles [MPS04] and stars [PS04]). 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).

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 both as a way to bound the number of values outside the set (forming the kernel) and as a tool in the color-coding dynamic programming (forming the algorithm). We are able to reduce the size of the constants inherent in the choice of hash functions in the original formulation of color-coding [AYZ95] 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 [Mar04,Woe03,CFJ04], our technique breaks new ground by refining these ideas in their joint use.

After introducing notation common to all the problems in Section 2, we first present the kernelization algorithm for r-Dimensional Matching (Section 3), followed by the fixed-parameter algorithm in Section 4. Next, we consider the modifications necessary to solve the other problems in Section 5. The paper concludes with a discussion of tuning the hash functions (Section 6) and further generalization of our techniques (Section 7).

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; in r-set packing, we find k disjoint sets; in graph packing, we find k vertex-disjoint subgraphs isomorphic to input H; and in graph edge packing, we find k edge-disjoint subgraphs isomorphic to input H. To unify the approach taken to the problems, we view each structure as an r-tuple (where in the last two cases $r = |V(H)|$ and $r = |E(H)|$, respectively); the problems differ in the ways the r-tuples must be disjoint and, in the case of graph packing, on additional constraints put on the structures.

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 example 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. The precise descriptions of the other problems will be deferred to Section 5.

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 $\mathsf{Fully\text{-}Reduce}(\mathcal{S}_I)$
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 $\mathcal{S}' =$*Reduce$(\mathcal{S},R)$ *for some R such that* 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.

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 $i \in \{j \mid R(j) = *\}$ 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 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 $\mathcal{P} - \{T\}$. Using the statement proved above, as $|\{j \mid R(j) = *\}| = \ell$ and $|\mathcal{P}-\{T\}| = k-1$, 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 can be proved by induction on the number of iterations; proofs for this and subsequent lemmas are omitted due to space limitations.

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 $\mu(\mathcal{S}',k) \neq \emptyset$.*

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 $x \in$* 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.*

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 $\mathsf{Kernel\text{-}Construct}(\mathcal{S})$
Input: A set $\mathcal{S} \subseteq \mathcal{A}$.
Output: A set $\mathcal{K} \subseteq \mathcal{S}$.
$\mathcal{S}' \leftarrow$Fully-Reduce(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) $|\mathcal{K}| \in O(k^r)$ and (b) $\mu(\mathcal{S},k) \neq \emptyset$ if and only if $\mu(\mathcal{K},k) \neq \emptyset$.*

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. Hence we propose a novel alternative, which combines the colour coding technique of Alon et al. [AYZ95] with the use of a maximal matching $\mathcal{M} \subseteq \mathcal{K}$.

In its original form, the colour-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 colours 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 colours X has size rk. The family of hash functions $\mathcal{F}$ used by Alon et al. is of size $2^{O(rk)}$ in our case. In Section 6 we will consider modifications towards improving this approach.

To modify colour-coding for our purposes, we first compute a maximal matching $\mathcal{M}$ in the kernel. By the maximality of $\mathcal{M}$, each r-tuple in a matching $\mathcal{P}$ of size k in the kernel must contain a value in $\mathcal{M}$. Hence there may be at most $(r-1)k$ values of $\mathcal{P}$ that do not belong to $\mathcal{M}$. We will seek a proper colouring of these values. Thus we choose a universe $U = \mathsf{val}(\mathcal{K}) - \mathsf{val}(\mathcal{M})$, and we set $|X| = (r-1)k$.

Our dynamic programming problem space has four dimensions. Two of them are the choice of hash function ϕ, and a subset W of values of $\mathcal{M}$ that might be used by $\mathcal{P}$. For each choice of these, there are two more dimensions associated with a possible subset $\mathcal{P}'$ of $\mathcal{P}$: the set of values Z in W it uses, and the set of colours C that ϕ assigns its values not in W. More formally, for $\mathcal{M}$ a maximal matching of $\mathcal{S} \subseteq \mathcal{A}$, then for any $W \subseteq \mathsf{val}(\mathcal{M})$, $\phi \in \mathcal{F}$, $Z \subseteq W$, and $C \subseteq X$ such that $|Z| + |C| \leq r \cdot |\mathcal{P}|$ and $|Z| + |C| \equiv 0 \bmod r$ we define

$$B_{\phi,W}(Z,C) = \begin{cases} 1 & \text{if there exists a matching } \mathcal{P}' \subseteq \mathcal{S} \text{ where } |\mathcal{P}'| = \frac{|Z|+|C|}{r}, \\ & \mathsf{val}(\mathcal{P}') \cap \mathsf{val}(\mathcal{M}) = Z, \text{ and } \phi(\mathsf{val}(\mathcal{P}') - Z) = C, \\ 0 & \text{otherwise} \end{cases}$$

where for convenience we use the notation $\phi(S)$ for a set S to be $\cup_{v \in S} \phi(v)$. In order to solve the problem, our goal is then to use dynamic programming to determine $B_{\phi,W}(Z,C)$ for each W, for each ϕ in the family $\mathcal{F}$, and for each Z and C such that $|Z| + |C| \leq rk$.

To count the number of dynamic programming problems thus defined, we note that there are at most $2^{(r-1)k}$ choices for W, C, and Z, and $2^{O(rk)}$ choices of $\phi \in \mathcal{F}$. Thus there are $2^{O(rk)}$ problems in total.

To formulate the recurrence for our dynamic programming problem, we note that $B_{\phi,W}(Z,C) = 1$ for $|Z| + |C| = 0$, and observe that $B_{\phi,W}(Z,C) = 1$, for $|Z| + |C| > 0$, holds precisely when there exists an r-tuple formed of a subset Z' of Z and a subset C' of C such that there is a matching of size one smaller using $Z - Z'$ and $C - C'$. For the ease of exposition, we define the function $P_\phi : 2^W \times 2^X \to \{0,1\}$ (computable in $O(k^r)$ time) as

$$P_\phi(Z',C') = \begin{cases} 1 & \text{if there exists an } r\text{-tuple } T \in \mathcal{S} \text{ where } Z' \subseteq \mathsf{val}(T) \\ & \text{and } \phi(\mathsf{val}(T) - Z') = C', \\ 0 & \text{otherwise} \end{cases}$$

Observing that each r-tuple must contain at least one element in Z, we can then calculate $B_{\phi,W}(Z,C)$ by dynamic programming as follows:

$$B_{\phi,W}(Z,C) = \begin{cases} 1 & \text{if there exist } Z' \subseteq Z,\ C' \subseteq C \text{ with } |Z'| \geq 1,\ |Z'| + |C'| = r, \\ & P_\phi(Z',C') = 1 \text{ and } B_{\phi,W}(Z - Z', C - C') = 1 \\ 0 & \text{otherwise} \end{cases}$$

One table entry can be computed in $O(k^r)$ time. The number of choices for Z' and C' can be bounded by $\binom{|Z|+|C|}{r} \leq \binom{rk}{r} = O(k^r)$. The algorithm below is a summary of the above description, where $\mathcal{F}$ is a set of hash functions from $\mathsf{val}(\mathcal{S}) - \mathsf{val}(\mathcal{M})$ to X.

In merging the following routine with the kernel construction, the maximal matching needs to be found only once. To find a maximal matching in the kernel, we use a greedy algorithm running in time $O(k^r)$. The running time is dominated by kernel construction (taking time $O(n)$) plus dynamic programming (taking time $2^{O(rk)}$). This yields the following theorem.

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

Routine $\mathsf{Color\text{-}Coding\text{-}Matching}(\mathcal{S})$
Input: A set $\mathcal{S} \subseteq \mathcal{A}$.
Output: A member of the set $\{\text{Yes}, \text{No}\}$.

Find a maximal matching $\mathcal{M}$ of $\mathcal{S}$.
If $|\mathcal{M}| \geq k$ then **output** "Yes" and **stop**.
set $V = \mathsf{val}(\mathcal{S}) - \mathsf{val}(\mathcal{M})$.
For each $W \subseteq \mathsf{val}(\mathcal{M})$, **do**
  for each coloring $\phi : V \to X$ where $\phi \in \mathcal{F}$
    for each $Z \subseteq W$ by increasing $|Z|$
      for each $C \subseteq X$ by increasing $|C|$, where $|Z| + |C| \equiv 0 \bmod r$
        compute $B_{\phi,W}(Z, C)$.
        if $B_{\phi,W}(Z, C) = 1$ and $|Z| + |C| = rk$, **output** "Yes" and **stop**.
Output "No".

5 Kernels and FPT Algorithms for the Other Problems

We can now define other problems addressable by our technique. To avoid introducing new notation, each of them will be defined in terms of sets of tuples, even though order within a tuple is not important in some cases. 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.

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 $|\mathcal{P}| \geq k$?

In order to define the graph problems, we define $G[S]$ to be the subgraph of G induced on the vertex set $S \subseteq V(G)$, namely the graph $G' = (V(G'), E(G'))$, where $V(G') = S$ and $E(G') = \{(u, v) \mid u, v \in S, (u, v) \in E(G)\}$. Moreover, we use $H \subseteq G$ to denote that H is a (not necessarily induced) subgraph of G.

Graph Packing
Instance: Two graphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$.
Parameter: A non-negative integer k.
Question: Does G contain at least k vertex-disjoint subgraphs each isomorphic to H, that is, for $\mathcal{S} \subseteq \mathcal{A} = V(G)^{|V(H)|}$, is there a subset $\mathcal{P}$ of $\mathcal{S}$ where for any $T, T' \in \mathcal{P}$, $\mathsf{val}(T) \cap \mathsf{val}(T') = \emptyset$, there is a subgraph $G' \subseteq G[\mathsf{val}(T)]$ that is isomorphic to H for any $T \in \mathcal{P}$, and $|\mathcal{P}| \geq k$?

Graph Edge-Packing
Instance: Two graphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$ such that H has no isolated vertices.
Parameter: A non-negative integer k.
Question: Does G contain at least k edge-disjoint subgraphs each isomorphic to H, that is, for $\mathcal{S} \subseteq \mathcal{A} = E(G)^{|E(H)|}$, is there a subset $\mathcal{P}$ of $\mathcal{S}$ where for any

$T, T' \in \mathcal{P}$, $\mathsf{val}(T) \cap \mathsf{val}(T') = \emptyset$, there is a subgraph G' that is isomorphic to H such that each edge of G' is in T for any $T \in \mathcal{P}$, and $|\mathcal{P}| \geq k$?

To obtain kernels for the above problems, we can adapt the ideas developed for r-Dimensional Matching. As in that case, 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. In the remainder of this section we detail the differences among the solutions for the various problems.

For each of these problems, 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. Since in the last two problems there is an additional constraint that the set of vertices or edges forms a graph isomorphic to H, the sets are forced to be of size exactly r; for r-Set Packing there is no such constraint, allowing smaller sets.

For all three problems, 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 $g(\ell, k) = r \cdot \ell \cdot (k-1) \cdot g(\ell-1, k) + 1$ for all $\ell > 0$, for $r = |V(H)|$ and $r = |E(H)|$ in the latter two problems. 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 or a maximal set of disjoint sets of vertices or edges yielding subgraphs isomorphic to H in the function Kernel-Construct. 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 $\mathsf{val}(\hat{T}) \cap \mathsf{val}(\tilde{T}) \neq \emptyset$. Since the r-tuples are in fact sets, positions have no meaning, and hence the number of such r-tuples will be the number of positions per tuple (that is, r) multiplied by the number of free positions (ℓ in any R' formed by fixing one more index) multiplied by the number of tuples in $\mathcal{P} - \{\mathcal{T}\}$ (that is, $k-1$) multiplied by $g(\ell-1, k)$. In total this gives us $r \cdot \ell \cdot (k-1) \cdot g(\ell-1, k) = g(\ell, k) - 1$ 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 problems r-Set Packing, Graph Packing, and Graph Edge-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 objects (disjoint sets, vertex-disjoint subgraphs isomorphic to H, or edge-disjoint subgraphs isomorphic to H, respectively) if and only if $\mathcal{K}$ has a set of at least k such objects.*

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

Our algorithms differ from that for r-Dimensional Matching only in the definitions of $B_{\phi,W}(Z,C)$ and P_ϕ. In particular, the conditions for $B_{\phi,W}(Z,C)$ to be 1 are as follows: there exists a set $\mathcal{P} \subseteq \mathcal{S}$ of disjoint sets where $|\mathcal{P}| \geq \frac{|Z|+|C|}{r}$, $\mathsf{val}(\mathcal{P}) \cap \mathsf{val}(\mathcal{M}) = Z$, and $\phi(\mathsf{val}(\mathcal{P}) - Z) = C$ (r-Set Packing); there exists a set $\mathcal{P} \subseteq \mathcal{S}$ such that $|\mathcal{P}| \geq \frac{|Z|+|C|}{r}$, for any $T, T' \in \mathcal{P}$, $\mathsf{val}(T) \cap \mathsf{val}(T') = \emptyset$, for each $T \in \mathcal{P}$ there is a subgraph $G' \subseteq G[\mathsf{val}(T)]$ that is isomorphic to T, $\mathsf{val}(\mathcal{P}) \cap \mathsf{val}(\mathcal{M}) = Z$, and $\phi(\mathsf{val}(\mathcal{P}) - Z) = C$ (Graph Packing); and there exists a set $\mathcal{P} \subseteq \mathcal{S}$ such that $|\mathcal{P}| \geq \frac{|Z|+|C|}{r}$, for any $T, T' \in \mathcal{P}$, $\mathsf{val}(T) \cap \mathsf{val}(T') = \emptyset$, for each $T \in \mathcal{P}$ there is a subgraph G' that is isomorphic to H such that each edge of G' is in T, $\mathsf{val}(\mathcal{P}) \cap \mathsf{val}(\mathcal{M}) = Z$, and $\phi(\mathsf{val}(\mathcal{P}) - Z) = C$ (Graph Edge-Packing). In addition, we alter the conditions for P_ϕ to be 1 in a similar manner, though no alteration is needed for r-Set Packing: for Graph Packing we add the condition that there is a subgraph $G' \subseteq G[\mathsf{val}(T)]$ that is isomorphic to H and for Graph Edge-Packing we add the condition that there is a subgraph G' that is isomorphic to H such that each edge of G' is in T.

The analysis of the algorithms depends on Lemma 6 and, in the graph problems the need to find all copies of H in G, resulting in the theorem below.

Theorem 2. *The problems r-Set Packing, Graph Packing, and Graph Edge-Packing can solved in time $O(n + 2^{O(rk)})$, $O(n^{|V(H)|} + 2^{O(rk)})$, and $O(n^{|E(H)|} + 2^{O(rk)})$, respectively.*

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 Section 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 colours, 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 colours 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 [SvEB85]. They give a scheme based on the pioneering work of Fredman, Komlós, and Szemerédi [FKS82] 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 colouring. A random $2|S|$-colouring 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.

7 Conclusions

Our results can be extended to handle problems such as packing or edge-packing graphs from a set of supplied graphs, and more generally to disjoint structures. To express all problems in a general framework, we view tuples as directed hypergraph edges, sets of tuples as hypergraphs, and inputs as triples of a sequence of hypergraphs, a matching-size parameter ℓ, and a supplied hypergraph G.

To define the necessary terms, we consider the projection of tuples and an associated notion of independence. Given a tuple $T = (v_1, \ldots, v_{r'})$ where $r' \leq r$ and a subset of indices $I = (i_1, \ldots, i_\ell) \subseteq \{1, \ldots, r\}$, we define the *projection of* T *on* I as $T[I] = (v_{i_1}, \ldots, v_m)$ where $m = \max\{1, \ldots, r'\} \cap \{i_1, \ldots, i_\ell\}$. To form a set of related tuples, given an hypergraph $\mathcal{S}$ and a subset $I \subseteq \{1, \ldots, r\}$ such that $|I| = \ell$, we denote as $w(\mathcal{S}, I) = \{T' \in V^\ell \mid \text{there exists a tuple } T \in \mathcal{S} \text{ such that } T[I] = T'\}$. We then say that a subsequence $\mathcal{C}' \subseteq \mathcal{C}$ is an *ℓ-matching* if for any pair of hypergraphs $\mathcal{S}, \mathcal{S}' \in \mathcal{C}'$, for all subsets of indices $I \subseteq \{1, \ldots, r\}$ of size ℓ, $w(\mathcal{S}, I) \neq w(\mathcal{S}', I)$.

Our problem is then one of finding a subsequence $\mathcal{C}'$ of the hypergraphs forming an ℓ-matching of size at least k such that the elements of each hypergraph in $\mathcal{C}'$ can be restricted to a smaller structure isomorphic to G. This problem subsumes all problems in this paper, and can be solved in time $O(n + 2^{O(\ell k)})$.

References

[ADP80] G. Ausiello, A. D'Atri, and M. Protasi. Structure preserving reductions among context optimization problems. *Journal of Computer and System Sciences*, 21:136–153, 1980.

[AYZ95] N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the Association for Computing Machinery*, 42(4):844–856, 1995.

[CFJ04] B. Chor, M. R. Fellows, and D. Juedes. 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)*, 2004.

[CFJK01] J. Chen, D. K. Friesen, W. Jia, and I. Kanj. Using nondeterminism to design deterministic algorithms. In *Proceedings of the 21st Conference on Foundations of Software Technology and Theoretical Computer Science (FST TCS)*, volume 2245 of *Lecture Notes in Computer Science*, pages 120–131. Springer, 2001.

[Coo71] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual Symposium on the Theory of Computing*, pages 151–158, 1971.

[Fel03] M. R. Fellows. 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)*, volume 2880 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2003.

[FHR+04] M. R. Fellows, P. Heggernes, F. A. Rosamond, C. Sloper, and J. A. Telle. 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)*, 2004.

[FKS82] M. L. Fredman, J. Komlós, and E. Szemerédi. Storing a sparse table with $o(1)$ worst-case access time. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 165–169, 1982.

[Hol81] I. Holyer. The NP-completeness of some edge-partition problems. *SIAM Journal on Computing*, 10(4):713–717, 1981.

[JZC04] W. Jia, C. Zhang, and J. Chen. An efficient parameterized algorithm for m-set packing. *Journal of Algorithms*, 50(1):106–117, 2004.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[Mar04] D. Marx. Parameterized complexity of constraint satisfaction problems. In *Proceedings of the 19th Annual IEEE Conference on Computational Complexity*, 2004.

[MPS04] L. Mathieson, E. Prieto, and P. Shaw. Packing edge disjoint triangles: a parameterized view. In *Proceedings of the International Workshop on Parameterized and Exact Computation*, 2004.

[PS04] E. Prieto and C. Sloper. Looking at the stars. In *Proceedings of the International Workshop on Parameterized and Exact Computation*, 2004.

[SvEB85] C. F. Slot and P. van Emde Boas. On tape versus core; an application of space efficient perfect hash functions to the invariance of space. *Elektronische Informationsverarbeitung und Kybernetik*, 21(4/5):246–253, 1985.

[Woe03] G. J. Woeginger. Exact algorithms for NP-hard problems, a survey. In *Combinatorial Optimization – Eureka, You Shrink!*, volume 2570 of *Lecture Notes on Computer Science*, pages 185–207. Springer, 2003.