Improved Algorithms for Path, Matching, and Packing Problems *

Jianer Chen
Department of Computer Science
Texas A&M University
College Station, TX 77843
chen@cs.tamu.edu

Songjian Lu
Department of Computer Science
Texas A&M University
College Station, TX 77843
sjlu@cs.tamu.edu

Sing-Hoi Sze
Department of Computer Science and
Department of Biochemistry & Biophysics
Texas A&M University
College Station, TX 77843
shsze@cs.tamu.edu

Fenghui Zhang
Department of Computer Science
Texas A&M University
College Station, TX 77843
fhzhang@cs.tamu.edu

ABSTRACT

We develop new and improved randomized and deterministic algorithmic techniques for PATH, MATCHING, and PACKING problems. Our randomized algorithms are based on a new divide-and-conquer technique, which leads to improved algorithms for these problems. For example, for the k-PATH problem, our randomized algorithm runs in time $O(4^k mk^{3.42})$ and space $O(nk \log k + m)$, improving the previous best randomized algorithm for the problem that runs in time $O(5.44^k km)$ and space $O(2^k kn + m)$. To achieve improved deterministic algorithms, we develop a lower bound $\Omega(2.718^k)$ and an improved upper bound $O(6.4^k n)$ on the number of k-colorings in a k-color coding scheme. This leads directly to a deterministic algorithm of time $O(12.8^k nm)$ for the k-PATH problem, improving the previous best deterministic algorithm for the problem that runs in time $O(c^k nm)$, where $c > 8000$. Our techniques also lead to similar or more significant improvements on randomized and deterministic algorithms for MATCHING and PACKING problems, such as 3-D MATCHING, 3-SET PACKING, and TRIANGLE PACKING.

1. Introduction

This paper studies new and improved algorithmic techniques for exact and parameterized algorithms for PATH, MATCHING, and PACKING problems that are NP-hard. This research direction has recently drawn considerable attention [1, 4, 7, 11, 13, 14, 16, 17, 19, 21].

The k-PATH problem (given a graph G and an integer k, either construct a simple path of k vertices in G or report no such a path exists) is closely related to a number of well-known NP-hard problems, such as the LONGEST PATH problem, the HAMILTONIAN PATH problem, and the TRAVELLING SALESMAN problem. Earlier algorithms [2, 17] for the k-PATH problem have running time bounded by $O(2^k k! n^{O(1)})$. Papadimitriou and Yannakakis [18] studied a restricted version of the problem, the $(\log n)$-PATH problem, and conjectured that the $(\log n)$-PATH problem can be solved in polynomial time. This conjecture was confirmed by Alon, Yuster, and Zwick [1], who presented for the k-PATH problem randomized and deterministic algorithms of running time $O(2^{O(k)} n^{O(1)})$. This also provides currently the best polynomial time approximation algorithm of ratio $O(n/\log n)$ for the LONGEST PATH problem. Very recently, the k-PATH problem has found applications in bioinformatics for detecting signaling pathways in protein interaction networks [21] and for biological subnetwork matchings [13].

The exact and parameterized MATCHING and PACKING problems were first studied in [6]. In particular, deterministic algorithms of running time $O(2^{O(k)}(3k)! n \log^4 n)$ were developed for the 3-D MATCHING problem (given a set S of triples and an integer k, either find a subset of k disjoint triples in S or report no such a subset exists) and the 3-SET PACKING problem (given a collection C of 3-sets and an integer k, either find a sub-collection of k disjoint 3-sets in C or report no such a sub-collection exists). The upper bounds for the complexity of the problems were subsequently improved to $O((5.7k)^k n)$ based on the *greedy localization* techniques [4, 11]. Koutis [14] developed randomized algorithms of time $O(10.88^{3k} n^{O(1)})$ and space $O(2^k)$, and deterministic algorithms of time $O(2^{O(k)})$ for the problems. This upper bound was further improved to $O((12.7D)^{3k})$ (where $D \geq 10.4$) by Fellows et al. [7]. Algorithms for packing a small subgraph in a given graph, such as TRIANGLE PACKING (given a graph G and integer k, either find a set of k vertex-disjoint triangles or report no such a set exists), have

*This work was supported in part by the National Science Foundation under the Grants CCR-0311590 and CCF-0430683.

also been studied [7, 16, 19].

Currently, the best randomized and deterministic algorithms for the k-PATH problem [1] and for the MATCHING and SET PACKING problems [14, 7] are all based on a technique called *color-coding* developed by Alon, Yuster, and Zwick [1]. Take the k-PATH problem as an example. We say that a simple path in a graph G is *properly colored* under a coloring of the vertices in G if no two vertices on the path are colored with the same color. The algorithms proposed in [1] proceed as follows. Suppose that there is a path P of k vertices in G, starting from a vertex v_0. To find the path P, first we somehow color the vertices of the graph G using k colors so that the path P is properly colored. Then we can use a (deterministic) dynamic programming algorithm, which for each vertex u records every possible color set C such that there is a properly colored simple path from v_0 to u that uses exactly the colors in the set C. Since there are at most 2^k different color sets, the dynamic programming algorithm runs in time $O(2^k km)$ and uses $O(2^k kn + m)$ space.

Therefore, the critical step is how to construct a coloring for the graph G so that the path P is properly colored. Alon, Yuster, and Zwick [1] proposed two approaches to this problem. The first is a randomized algorithm of running time $O(e^k n)$ that produces $O(e^k)$ colorings for the graph G in which with high probability at least one coloring properly colors the path P. The second is a deterministic algorithm based on the hashing schemes studied by Fredman, Komlos, and Szemeredi [8] and Schmidt and Siegel [20], which constructs a set of $O(c^k n^{O(1)})$ colorings for the graph G in which at least one colors the path P properly, where $c >> 1$ is a constant. This, plus the above dynamic programming algorithm, gives for the k-PATH problem a randomized algorithm of running time $O((2e)^k n^{O(1)}) = O(5.44^k n^{O(1)})$ and space $O(2^k kn+m)$ and a deterministic algorithm of running time $O((2c)^k n^{O(1)})$. The currently best randomized and deterministic algorithms for MATCHING and SET PACKING [14, 7] follow the same principle: first we color the elements so that no two elements in the subset of our interests are colored with the same color, then we apply a deterministic algorithm (e.g., dynamic programming) on the set of colored elements to search for the interested subset.

This method is of great theoretical importance. In particular, it confirms Papadimitriou and Yannakakis's conjecture that the $(\log n)$-PATH problem can be solved in polynomial time. On the other hand, both the time complexity and the space complexity of the process are quite high. Actually it can be verified that for the deterministic algorithm of time $O((2c)^k kn^{O(1)})$ for the k-PATH problem described in [1], the constant c is at least 4000 (a more detailed analysis on the algorithm will be given in later discussions). In consequence, the deterministic algorithm proposed in [1] has its running time of the form $O(d^k n^{O(1)})$, where $d > 8000$. Obviously, such an algorithm will quickly become impractical even for very small values of k. Moreover, the space complexity of all the randomized algorithms described above for PATH, MATCHING, and PACKING is exponential in k, which is also remarkable.

In this paper, we study new techniques to develop improved randomized and deterministic algorithms for the above PATH, MATCHING, and PACKING problems. Our first result is a randomized divide-and-conquer method. Roughly speaking, suppose that we are looking for a subset S_k of k elements in a universal set S. We first randomly partition the subset S_k into two halves of equal size, then recursively look for a subset of $k/2$ elements in each of the halves. This simple method directly leads to improved randomized algorithms. For the k-PATH problem, this new method gives a randomized algorithm of time $O(4^k mk^{3.42})$ and space $O(nklogk + m)$, improving the previous best randomized algorithm for the problem of time $O(5.44^k km)$ and space $O(2^k kn + m)$ [1]. For the 3-D MATCHING and 3-SET PACKING problems, the method gives randomized algorithms of time $O(2.52^{3k} n)$ and space $(nk \log k)$, improving the previous best randomized algorithms for the problems of time $O(10.88^{3k} n^{O(1)})$ and space $O(2^{3k})$ [14].

Refs.	k-path	3-D match	3-set pack	Δ-pack
[1]	$> 8000^k$			
[14]		$> 32000^{3k}$	$> 32000^{3k}$	
[7]*		$(12.7D)^{3k}$	$(12.7D)^{3k}$	$(12.7D)^{3k}$
Ours	12.8^k	12.8^{3k}	12.8^{3k}	12.8^{3k}
Ours+[15]		2.80^{3k}	4.68^{3k}	4.68^{3k}

* In this paper, Fellows et al. developed a $13k$-color coding scheme of size 12.7^{3k}, and left the remaining procedures for the algorithms un-described. Here we have used $O(D^{3k})$ to denote the complexity of searching for a subset of $3k$ symbols in a set colored with $13k$ colors. A straightforward implementation of this process will have $D \geq 10.4$.

Figure 1: Comparison of deterministic algorithms

To develop improved deterministic algorithms for the PATH, MATCHING, and PACKING problems, we study systematically the complexity of *k-color coding schemes*. A *k-coloring* of a set S is a mapping from S to $\{0, \ldots, k-1\}$. A collection $\mathcal{C}$ of k-colorings of S is a *k-color coding scheme* for S if for each subset S_k of k elements in S there is a k-coloring F in $\mathcal{C}$ such that no two elements in S_k are colored with the same color under F. Denote by $\tau(n, k)$ the minimum size of a k-color coding scheme for a set of n elements. We study upper and lower bounds for the function $\tau(n, k)$. First, we develop a lower bound and prove that $\tau(n, k) > 2.718^k$. To improve the upper bound for $\tau(n, k)$, we propose a new four-level hashing procedure that uses the techniques of kernelization, collision minimization, and recursive construction. The kernelization technique used in the first level of the procedure is based on the hashing function studied in [8]. The next two levels in the procedure use a hashing function that is nearly optimal in terms of collision minimization. The last level of the procedure comes from a non-trivial recursive formula for $\tau(n, k)$ and careful constructions of k-color coding schemes for small values of k. These techniques induce a new k-color coding scheme of size $O(6.4^k n)$. This is much better than the previous upper bound for $\tau(n, k)$, which is larger than 4000^k [1].

The improved upper bound on $\tau(n, k)$ directly induces significant improvements on deterministic algorithms for the PATH, MATCHING, and PACKING problems. The comparison of the previous best algorithms and the improved algorithms based on our new k-color coding scheme is given in Figure 1.

We make a few remarks on our results before we proceed to technical discussions. Many NP-hard problems are concerned with searching for a subset S_k of k elements in a given set S such that the subset S_k satisfies certain properties. The k-color coding schemes seem to provide a convenient "pre-classification" of the elements in S so that all elements in the subset S_k are colored with distinct colors, which will significantly narrow down the search space for the problem. From this point of view, the study of k-color

find-paths(G', P', k)
input: G' a subgraph of G, P' a set of k'-paths in G that contain no vertex in G', an integer $k \geq 1$;
output: a set P of paths, each is a concatenation of a k'-path in P' and a k-path in G';
1. $P = \emptyset$;
2. **if** $k = 1$ **then**
   if $P' = \emptyset$ **then return** all 1-paths in G';
   else for each (u, k')-path p in P' and each vertex w in G'
   if (u, w) is an edge in G **then**
   concatenate p and w to make a $(w, k'+1)$-path p';
   add p' to P if no $(w, k'+1)$-path is in P;
   return P;
3. **loop** $2.51 \cdot 2^k$ times
3.1. randomly partition the vertices in G' into two parts V_L and V_R;
3.2. let G_L and G_R be the subgraphs induced by V_L and V_R, respectively;
3.3. $P_L =$ **find-paths**$(G_L, P', \lceil k/2 \rceil)$;
3.4. **if** $P_L \neq \emptyset$ **then**
3.5. $P_R =$ **find-paths**$(G_R, P_L, k - \lceil k/2 \rceil)$;
3.6. **for** each $(u, k'+k)$-path p in P_R
3.7. add p to P if no $(u, k'+k)$-path is in P;
4. **return** P;

Figure 2: A divide-and-conquer algorithm

coding schemes seems to be of general interests in solving NP-hard problems.

We also remark on how significant our improvement for the upper bound for $\tau(n, k)$ is. All functions 5.44^k, 4^k, 4000^k, and 6.4^k are of the form $2^{O(k)}$. However, these functions are not linearly related. In fact, for algorithms whose running time is of the form $O(c^k n^{O(1)})$ where c^k is the dominating term (as this is the case for many NP-hard problems), a small reduction on the constant c will result in significant improvement on the running time. In particular, the value 4000^k is larger than the fourth power of the value 6.4^k.

2. Randomized divide-and-conquer

In this section, we describe a new randomized divide-and-conquer method, which will induce improved randomized algorithms for a number of PATH, MATCHING, and PACKING problems. To make our discussion more specific, we will describe the method in detail based on the k-PATH problem. We then explain briefly how the method is applied to MATCHING and PACKING problems. Throughout this paper, we will denote by $e = 2.718\cdots$ the base of the natural logarithm.

The randomized algorithm for k-PATH is given in Figure 2. A simple path in a graph G is a (u, k)-*path* if it contains exactly k vertices and if one end of the path is u. In particular, for any vertex u, a $(u, 1)$-path consists of a single vertex u. When the vertex u is irrelevant, a (u, k)-path will be simply called a k-*path*. Our algorithm **find-paths**(G', P', k) on the subgraph G' and a set P' of k'-paths (where no vertex on the paths in P' is in G') returns a set P of paths, each is a concatenation of a k'-path in P' and a k-path in G' (if no such paths exist, the algorithm returns an empty set). In particular, the algorithm **find-paths**$(G', \emptyset, k)$ returns a set of k-paths in the graph G'.

LEMMA 2.1. *For integer $k > 1$, $\lceil \log k \rceil = \lceil \log(\lceil \frac{k}{2} \rceil) \rceil + 1$.*

THEOREM 2.2. *On a graph $G = (V, E)$ with n vertices and m edges and an integer $k \geq 1$, if the graph G contains a (u, k)-path for a vertex u, then with probability larger than $1 - 1/e > 0.632$, the set P returned by the algorithm* **find-paths**$(G, \emptyset, k)$ *contains a (u, k)-path. The algorithm* **find-paths**$(G, \emptyset, k)$ *runs in time $O(4^k m k^{3.42})$ and in space $O(nk \log k + m)$.*

PROOF. To prove the first part, we prove the following claims using induction on k:

1. If $P' = \emptyset$ and G' has a (u, k)-path, then with probability larger than $1 - 1/e$, the set P returned by the algorithm **find-paths**(G', P', k) includes a (u, k)-path.

2. If $P' \neq \emptyset$ and G' has a (u, k)-path whose other end is connected to an end vertex of a path in P', then with probability larger than $1 - 1/e$, the set P returned by the algorithm **find-paths**(G', P', k) contains a $(u, k' + k)$-path.

By the algorithm, the claims are obviously true for $k = 1$. Now let $k > 1$. First consider the case when $P' = \emptyset$. Suppose that

$$[u_1, u_2, \ldots, u_{k_1}, u_{k_1+1}, \ldots, u_k]$$

is a (u_k, k)-path in G', where $k_1 = \lceil k/2 \rceil$. Then with probability $1/2^k$, step 3.1 of the algorithm puts vertices $u_1, u_2, \ldots, u_{k_1}$ into V_L, and vertices $u_{k_1+1}, \ldots, u_k$ into V_R. If this is the case, then the graph G_L contains the (u_{k_1}, k_1)-path $p_0 = [u_1, \ldots, u_{k_1}]$, and the graph G_R contains the $(u_k, k - k_1)$-path $p_1 = [u_{k_1+1}, \ldots, u_k]$. By the inductive hypothesis, with probability larger than $1 - 1/e$, P_L obtained from step 3.3 includes a (u_{k_1}, k_1)-path. Now the $(u_k, k - k_1)$-path $[u_{k_1+1}, \ldots, u_k]$ in G_R has its other end u_{k_1+1} connected to the (u_{k_1}, k_1)-path in P_L. Therefore with probability larger than $1 - 1/e$, P_R obtained in step 3.5 contains a path of length $k_1 + (k - k_1) = k$ that ends with u_k, i.e., a (u_k, k)-path. Therefore in each loop of step 3, the probability ρ that a (u_k, k)-path is added to the set P is larger than

$$\frac{(1 - 1/e)^2}{2^k} > \frac{0.6322}{2^k} > \frac{1}{2.51 \cdot 2^k}.$$

When $P' \neq \emptyset$, we follow the same argument as before except that we require that the (u_k, k)-path in G' has its other end connected to the end of a k'-path in P'. So P_L contains a $(u_{k_1}, k' + k_1)$-path p that is a concatenation of a k'-path in P' and a k_1-path in G_L, and P_R contains a $(u_k, k' + k)$-path that is a concatenation of a $(k' + k_1)$-path in P_L and a $(k - k_1)$-path in G_R.

Since step 3 of the algorithm loops $2.51 \cdot 2^k$ times, the overall probability that the algorithm returns a set of paths that contains a (u_k, k)-path (when the set P' is empty) or a $(u_k, k' + k)$-path (when the set P' is not empty) is:

$$1 - (1 - \rho)^{2.51 \cdot 2^k} > 1 - \left(1 - \frac{1}{2.51 \cdot 2^k}\right)^{2.51 \cdot 2^k} > 1 - \frac{1}{e}.$$

This proves the first part of the theorem.

To analyze the time complexity, let $T(k)$ be the running time of the algorithm **find-paths**(G', P', k). Without loss of generality, we can assume $m \geq n$. From the algorithm, we get the following recurrence relation:

$$T(k) = 2.51 \cdot 2^k (cm + T(\lceil k/2 \rceil) + T(k - \lceil k/2 \rceil)), \qquad (1)$$

where $c > 0$ is a constant. We claim that for all $k > 0$,

$$T(k) \le c \cdot (10.7)^{\lceil \log k \rceil} m 2^{2k}, \qquad (2)$$

and we prove it by induction on k. Obviously $T(1) \le cm$ if c is a sufficiently large constant, so inequality (2) holds for $k = 1$. Let $k > 1$, then:

$$\begin{aligned} T(k) &= 2.51 \cdot 2^k \cdot (cm + T(\lceil k/2 \rceil) + T(k - \lceil k/2 \rceil)) \\ &\le 2.51 \cdot 2^k \cdot \left(cm + 2c \cdot (10.7)^{\lceil \log \lceil k/2 \rceil \rceil} m 2^{2\lceil k/2 \rceil} \right) \\ &\le 2.51 \cdot 2^k \cdot \left(cm + \frac{2c}{10.7} \cdot (10.7)^{\lceil \log \lceil k/2 \rceil \rceil + 1} m 2^{k+1} \right) \\ &= c \cdot (10.7)^{\lceil \log k \rceil} m 2^{2k} \cdot 2.51 \cdot \left(\frac{1}{10.7^{\lceil \log k \rceil} 2^k} + \frac{4}{10.7} \right) \\ &\le c \cdot (10.7)^{\lceil \log k \rceil} m 2^{2k} \cdot 2.51 \cdot (1/(10.7 \cdot 4) + 4/10.7) \\ &< c \cdot (10.7)^{\lceil \log k \rceil} m 2^{2k}. \end{aligned}$$

Here in the second step of the above derivation, we have used $k - \lceil k/2 \rceil \le \lceil k/2 \rceil$, in the third step, we have used $2\lceil k/2 \rceil \le k + 1$, and in the fourth step we have used Lemma 2.1. Thus, the running time $T(k)$ of the algorithm **find-paths**$(G, \emptyset, k)$ is $O((10.7)^{\lceil \log k \rceil} m 2^{2k}) = O(4^k m k^{3.42})$.

In terms of space complexity, each recursive call to the algorithm **find-paths** uses space $O(nk)$ (mainly for the sets P_L, P_R, and P). Since on a graph G and an integer k, the recursive depth of the algorithm is $O(\log k)$, we conclude that the space complexity of the algorithm **find-paths**$(G, \emptyset, k)$ is $O(nk \log k + m)$. □

Using Theorem 2.2, we can develop $O(4^k m k^{3.42})$ time randomized algorithms of arbitrarily small error bound for the k-PATH problem. For example, to achieve an error bound 0.0001, we can run the algorithm in Theorem 2.2 t times, where t satisfies $(1/e)^t \le 0.0001$ (e.g., $t = 10$).

We compare our algorithm in Theorem 2.2 with previously known algorithms for the k-PATH problem. To our knowledge, there are two kinds of randomized algorithms for the k-PATH problem. The first kind is based on random permutations of graph vertices followed by searching for a k-path in a directed acyclic graph (see [1, 13] for details). The algorithm runs in $O(mk!)$ time and $O(m)$ space. The second kind is proposed by Alon, Yuster, and Zwick [1] based on random coloring of graph vertices followed by a dynamic programming searching for a k-path in the colored graph. The algorithm runs in time $O((2e)^k km) = O(5.44^k km)$ and space $O(2^k kn + m)$ (the space is mainly for the dynamic programming phase). Compared to these algorithms, our algorithm has a significantly improved running time and runs in polynomial space. In fact, if we only need to know if the graph has a k-path or not, a slight modification of our algorithm can reduce the space complexity to $O(n \log k + m)$.

Remark. It seems that we have to be more careful when we analyze an exponential time algorithm based on the divide-and-conquer method. Certain common techniques from traditional algorithm analysis seem not directly applicable. For example, we cannot simply assume that the parameter k is a power of 2 since the extension from this special case to the case for general k does not seem to give the same complexity bound. In fact, in the case when k is a power of 2, it is quite trivial to verify (using induction) that $T(k) \le O(4^k m k^{2.52})$. However, it seems not easy to extend this bound to the case for general k.

The above randomized divide-and-conquer method can also be used to develop improved algorithms for MATCHING and PACKING problems. Consider the 3-D MATCHING problem (given a set S of triples and an integer k, either find a subset S_k of k disjoint triples or report no such a subset exists). In the case when such a subset S_k exists, let A_k be the set of $3k$ symbols in the triples in S_k. With a probability $\binom{k}{k/2}/2^{3k} = O(1/(2^{2k}\sqrt{k}))$, we can partition the symbols in A_k into two subsets A'_k and A''_k such that A'_k contains the $3k/2$ symbols in $k/2$ triples in S_k and A''_k contains the $3k/2$ symbols in the other $k/2$ triples in S_k. Now the set S of triples can be partitioned into two subsets S' and S'' in terms of A'_k and A''_k, and a subset of $k/2$ triples is searched recursively in each of the sets S' and S''. An analysis similar to that in Theorem 2.2 shows that this algorithm runs in time $O(2.52^{3k} n)$ and space $O(nk \log k)$ and finds the subset of k triples with high probability. It is straightforward to modify this algorithm to obtain an algorithm of the same time and space complexity for the 3-SET PACKING problem (given a collection of 3-sets and an integer k, either find a sub-collection of k disjoint 3-sets or report no such a sub-collection exists).

THEOREM 2.3. *The* 3-D MATCHING *and* 3-SET PACKING *problems can be solved by randomized algorithms in time* $O(2.52^{3k} n)$ *and space* $O(nk \log k)$.

Note that the previous best randomized algorithms for the problems [14] take time $O(10.88^{3k} n^{O(1)})$ and space $O(2^{3k} n^{O(1)})$, where the space is exponential in k.

3. Color coding schemes: preliminary and lower bounds

For the rest of this paper, we study deterministic algorithms for PATH, MATCHING, and PACKING problems. First we study the method of *color coding* proposed by Alon, Yuster, and Zwick [1] and develop a new color coding scheme that significantly improves the complexity of previous color coding schemes. The new color coding scheme then directly induces significantly improved algorithms.

Let S be any set and let W be a subset of S. A function f on S is *injective* from W if for any two different elements x and y in W, we have $f(x) \ne f(y)$. For each integer m, denote by Z_m the integer set $\{0, 1, \ldots, m-1\}$. In particular, if m is a prime number, then Z_m is a field under addition and multiplication modulo m.

Definition A *k-coloring* of a set S is a function from S to Z_k. A collection $\mathcal{F}$ of k-colorings of S is a *k-color coding scheme* for S if for any subset W of k elements in S, there is a k-coloring f_W in $\mathcal{F}$ that is injective from W. The *size* of the k-color coding scheme $\mathcal{F}$ is equal to the number of k-colorings in $\mathcal{F}$.

Fix integers n and k, where $n \ge k$. We will concentrate on k-color coding schemes for the set Z_n. Denote by $\tau(n, k)$ the minimum size of a k-color coding scheme for the set Z_n.

THEOREM 3.1. *If Z_{k^2} has a k-color coding scheme of size r, then Z_n has a k-color coding scheme of size at most* $2nr$.

PROOF. By Bertrand's Conjecture [10], there is a prime number p satisfying $n \le p < 2n$.

For each integer $a \in Z_p$, define a function ψ_a from Z_n to Z_{k^2} by

$$\psi_a(x) = (ax \bmod p) \bmod k^2.$$

Fredman, Komlos, and Szemeredi [8] proved that for each subset W of k elements in Z_n, there is an $a \in Z_p$ such that the function ψ_a is injective from W.

Let $\mathcal{F}_{k^2} = \{F_1, F_2, \ldots, F_r\}$ be a k-color coding scheme of size r for Z_{k^2}. Consider the following collection of k-colorings of Z_n:

$$\mathcal{F}_n = \{F_i \circ \psi_a \mid 1 \le i \le r, \ \ a \in Z_p\}$$

where $F_i \circ \psi_a$ is a function from Z_n to Z_k defined by $(F_i \circ \psi_a)(x) = F_i(\psi_a(x))$. The collection $\mathcal{F}_n$ contains $rp \le 2nr$ k-colorings of Z_n. Moreover, for any subset W of k elements in Z_n, let b be the element in Z_p such that the function ψ_b is injective from W. Thus the image of W under the function ψ_b is a subset W' of k elements in Z_{k^2}. Since $\mathcal{F}_{k^2}$ is a k-color coding scheme for Z_{k^2}, there is a k-coloring F_j in $\mathcal{F}_{k^2}$ such that F_j is injective from W'. Now it is easy to verify that the function $F_j \circ \psi_b$ in $\mathcal{F}_n$ is injective from W. Since W is an arbitrary subset of k elements in Z_n, we conclude that $\mathcal{F}_n$ is a k-color coding scheme of size at most $2nr$ for Z_n. □

COROLLARY 3.2. $\tau(n,k) \le 2n \cdot \tau(k^2,k)$.

Therefore, the values $\tau(n,k)$ and $\tau(k^2,k)$ differ by at most a polynomial factor. This "kernelization" result enables us to concentrate on the study of the values $\tau(k^2,k)$.

In the following we derive a lower bound for $\tau(n,k)$.

THEOREM 3.3. *For any real number $\epsilon > 0$, the value $\tau(n,k)$ is larger than $(e-\epsilon)^k$ when n is sufficiently large.*

PROOF. Let F be any k-coloring of the set Z_n. Suppose that F maps n_0 elements in Z_n to 0, n_1 elements in Z_n to 1, ..., and n_{k-1} elements in Z_n to $k-1$, where $n_0 + n_1 + \cdots + n_{k-1} = n$. It is easy to see that the number of subsets of k elements in Z_n from which the function F is injective is equal to

$$n_0 \cdot n_1 \cdot \cdots \cdot n_{k-1} \le (n/k)^k.$$

On the other hand, the total number of subsets of k elements in Z_n is equal to $\binom{n}{k}$. Therefore, the number of k-colorings in a k-color coding scheme for the set Z_n is at least $\binom{n}{k}/(n/k)^k$. Using Stirling's approximation [9], we have

$$\binom{n}{k} = \Omega\left(\frac{n^n}{\sqrt{k}(n-k)^{n-k}k^k}\right).$$

Therefore, the number of k-colorings in a k-color coding scheme is at least of order

$$\frac{n^{n-k}}{\sqrt{k}(n-k)^{n-k}} = \frac{\left(1+\frac{k}{n-k}\right)^{n-k}}{\sqrt{k}} = \frac{1}{\sqrt{k}}\left[\left(1+\frac{k}{n-k}\right)^{\frac{n-k}{k}}\right]^k.$$

For any real number $\epsilon > 0$, this value is larger than $(e-\epsilon)^k$ when n is sufficiently large. □

As suggested by Alon, Yuster, and Zwick [1], a k-color coding scheme for Z_n can be constructed based on the classic study on perfect hashing functions. In particular, they suggested the class of hashing functions proposed by Fredman, Komlos, and Szemeredi [8] and refined by Schmidt and Siegel [20]. The hashing function class $\mathcal{F}$ described in [20] has the following property: for any subset W of k elements in Z_n, there is a hashing function f_W in $\mathcal{F}$ that is perfect (in our language, f_W is injective from W). Therefore, the hashing function class $\mathcal{F}$ is a k-color coding scheme for Z_n. Moreover, Schmidt and Siegel [20] described in detail how the hashing function class $\mathcal{F}$ is constructed, and showed that each of the hashing functions in $\mathcal{F}$ can be encoded in $\log n + ck$ binary bits, where $c \ge 12$.[1]

Therefore, the k-color coding scheme described in [20] is of size $2^{\log n + ck} = n(2^c)^k$, which is larger than 4000^k if we ignore the polynomial terms. So far this is the best upper bound on the value $\tau(n,k)$. On the other hand, our Theorem 3.3 above shows that there is a constant c_1, $c_1 > 2.718$, such that $\tau(n,k) = \Omega(c_1^k)$. Therefore, the dominating factor in the size $\tau(n,k)$ of k-color coding schemes for the set Z_n seems to be of form c_0^k for a constant c_0, where $2.718 < c_0 \le 4000$. It is interesting to investigate the precise value of c_0. In the next two sections, we present a new k-color coding scheme that significantly improves the upper bound for $\tau(n,k)$. More specifically, we show that $c_0 \le 6.4$.

4. Color coding schemes for small k

We first study the properties of color coding schemes, from which we will be able to derive upper bounds for the value $\tau(n,k)$ for small n and k. Due to the space limit, all proofs in this section are given in the Appendix.

LEMMA 4.1. *For any positive integers n and k, $n \ge k$, there is a k-color coding scheme of size $\binom{n}{k}$ for the set Z_n.*

For the case $k \le 2$ and $k = n$, we have

LEMMA 4.2. *For any $n \ge 2$, $\tau(n,0) \le 1$, $\tau(n,1) = 1$, $\tau(n,n) = 1$, and $\tau(n,2) \le \lceil \log n \rceil$.*

For general k, we have the following recurrence relation.

LEMMA 4.3. *If $n = n_1 + \cdots + n_r$, where all $n_j \ge 1$, then*

$$\tau(n,k) \le \sum_{0 \le k_1 \le n_1, \ldots, 0 \le k_r \le n_r}^{k_1 + \cdots + k_r = k} \left(\frac{\tau(\#[k_j \le 1], \#[k_j = 1])}{\binom{\#[k_j \le 1]}{\#[k_j = 1]}} \prod_{k_j \ge 2} \tau(n_j, k_j) \right),$$

where $\#[k_j \le 1]$ and $\#[k_j = 1]$ denote the numbers of k_j's in the list $[k_1, \ldots, k_r]$ such that $k_j \le 1$ and $k_j = 1$, respectively.

By Lemma 4.2 and Lemma 4.3, for small values of n and k, we can derive specific upper bounds $\tau_0(n,k)$ for the values $\tau(n,k)$ and construct k-color coding schemes for the set Z_n. We first did this for very small values of n and k. Then using a computer program and based on Lemma 4.3, we also did this for larger values of n and k. The values $\tau_0(n,k)$ for these small n and k help us to prove the following lemma.

[1]The hashing functions described in [20] are encoded using at least the following binary strings: a binary string of $\log n$ bits, a binary string T_0 of $4k$ bits, a binary string T_1 of k bits, a binary string T_2 of $4k$ bits, and a binary string K_0 of $3k$ bits. Strictly speaking, the hashing function class given in [20] is weaker than what we described: with the binary strings given above, the hashing function in $\mathcal{F}$ that is injective from a subset W of k elements in Z_n only maps Z_n into Z_{3k} instead of Z_k.

LEMMA 4.4. *There is a collection $\{\mathcal{F}_2, \mathcal{F}_3, \ldots\}$ of color coding schemes, where $\mathcal{F}_k$ is a k-color coding scheme of size $\tau_0(k(k-1), k)$ for the set $Z_{k(k-1)}$, such that for any list of non-negative integers $[k_1, k_2, \ldots, k_r]$ satisfying $k_1 + \cdots + k_r = k$ and $\sum_{j=1}^{r} k_j(k_j - 1) \leq 4k$, we have $\prod_{k_j \geq 2} \tau_0(k_j(k_j - 1), k_j) \leq 2.4142^k$.*

5. A new color coding scheme

In this section, we develop a new k-color coding scheme for the set Z_n. According to Corollary 3.2, we can concentrate on the case $n = k^2$. Therefore, throughout this section, we assume that $n = k^2$. Moreover, we will first consider the case that k is divisible by 4 and let $k' = k/4 - 1$. The case for general k will be handled after the discussion for this special case.

5.1 A k-coloring algorithm for k divisible by 4

Let p be a prime number satisfying $n \leq p < 2n$ (such a prime number exists by Bertrand's Conjecture [10]). The prime number p can be obtained in time $O(n\sqrt{n}) = O(k^2)$ using the straightforward primality testing algorithm.

We first present a k-coloring algorithm for the set Z_n. Our k-coloring algorithm is associated with a set of parameters satisfying the following properties:

C1 A pair of integers (a, b), where $0 < a \leq p-1$, $0 \leq b \leq p-1$;

C2 A list $C = [c_0, c_1, \ldots, c_{k'}]$ of non-negative integers, where $k' = k/4 - 1$, $\sum_{j=0}^{k'} c_j = k$, and $\sum_{j=0}^{k'} c_j(c_j - 1) \leq 4k$. Let $C_{>1}$ be the sublist of C by removing all $c_j \leq 1$;

C3 A list $L = [(a_1, b_1), (a_2, b_2), \ldots, (a_r, b_r)]$ of pairs of integers, where $0 < a_i \leq p-1$, $0 \leq b_i \leq p-1$, and $r \leq \log |C_{>1}|$;

C4 A mapping from the elements in the list $C_{>1}$ to the elements in the list L such that at least half of the c_j's in $C_{>1}$ are mapped to (a_1, b_1), at least half of the c_j's that are not mapped to (a_1, b_1) are mapped to (a_2, b_2), at least half of the c_j's that are not mapped to (a_1, b_1) and (a_2, b_2) are mapped to (a_3, b_3), and so on.

C5 A list of colorings $[F_{c_0}, F_{c_1}, \ldots, F_{c_{k'}}]$, where for $c_j > 1$, F_{c_j} is a c_j-coloring of the set $Z_{c_j(c_j-1)}$, taken from the c_j-color coding scheme $\mathcal{F}_{c_j}$ given in Lemma 4.4 (for $c_j \leq 1$, F_{c_j} is irrelevant).

We also define a function as follows. For three given integers s, a, b, where $1 < s < n$, $0 < a \leq p-1$, and $0 \leq b \leq p-1$, define a function $\phi_{a,b,s}$ from the set Z_n to the set Z_s by

$$\phi_{a,b,s}(x) = ((ax + b) \bmod p) \bmod s. \quad (3)$$

Our k-coloring algorithm is given in Figure 3. Since $\sum_{j=0}^{k'} c_j = k$, it is easy to verify that the algorithm **Coloring** produces a k-coloring for the set Z_n.

For each different set of parameters satisfying conditions **C1**-**C5**, the algorithm **Coloring** may produce a different k-coloring for the set Z_n. We first consider the number of different combinations of these parameters satisfying conditions **C1**-**C5**.

Coloring
input: parameters as specified in **C1**-**C5**;
output: a k-coloring of the set Z_n;
1. **for** $j = 0$ **to** $k' = k/4 - 1$ **do**
   $U_j = \{x \mid x \in Z_n,\ \phi_{a,b,k/4}(x) = j\}$;
2. **for** each U_j such that $c_j > 1$, suppose that c_j is mapped to (a_i, b_i) **do**
   for $t = 0$ **to** $c_j(c_j - 1) - 1$ **do**
   $U_{j,t} = \{x \mid x \in U_j,\ \phi_{a_i,b_i,c_j(c_j-1)}(x) = t\}$;
   create c_j new colors $\tau_{j,0}$, $\tau_{j,1}$, $\ldots$, τ_{j,c_j-1};
   assign all elements in $U_{j,t}$ with color $\tau_{j,s}$ if the c_j-coloring F_{c_j} for $Z_{c_j(c_j-1)}$ assigns color s to the element t;
3. for each $c_j = 1$, create a new color τ_j and assign all elements in U_j the color τ_j;
4. assign all elements in $\bigcup_{c_j=0} U_j$ arbitrarily using the colors created in steps 2-3.

Figure 3: A coloring algorithm

THEOREM 5.1. *The total number of combinations of the parameters satisfying conditions* **C1**-**C5** *is bounded by $O(6.383^k k^{\log k - 4})$, and these combinations can be enumerated systematically.*

PROOF. There are $p^2 = O(n^2) = O(k^4)$ pairs of integers (a, b) satisfying condition **C1**.

Consider condition **C2**. We represent each list $C = [c_0, \ldots, c_{k'}]$ satisfying condition **C2** using a single binary string B_C of length $5k/4 - 1$ in which there are exactly $k/4 - 1$ 0-bits. The $k/4 - 1$ 0-bits in B_C divide B_C into $k/4$ "segments" such that the j-th segment contains exactly c_j 1-bits (in particular, the segment between two consecutive 0's in B_C corresponds to a $c_j = 0$). It is easy to verify that any list C satisfying condition **C2** is uniquely represented by such a binary string B_C. Note that the number of binary strings of length $5k/4-1$ with exactly $k/4-1$ 0-bits is equal to $\binom{5k/4-1}{k/4-1} \leq 1.8692^k$. We conclude that the total number of different lists satisfying condition **C2** is bounded by 1.8692^k. Note that all these lists can be systematically enumerated based on the binary string representation described above.

Since $r \leq \log(|C_{>1}|) \leq \log(k/4) = \log k - 2$, there are at most $\log k - 2$ pairs in each list L satisfying condition **C3**. By condition **C3**, there are $p^2 = O(k^4)$ possible pairs for each (a_i, b_i). Therefore, the total number of lists L satisfying condition **C3** is bounded by $O(k^{4 \log k - 8})$.

Now we discuss when a list $C = [c_0, \ldots, c_{k'}]$ satisfying condition **C2** and a list $L = [(a_1, b_1), \ldots, (a_r, b_r)]$ satisfying condition **C3** are given, how many different mappings from $C_{>1}$ to L can be there that satisfy condition **C4**. Let $q = |C_{>1}| \leq k/4$. We use a binary string $A_{>1}$ to represent a mapping from $C_{>1}$ to L. The binary string $A_{>1}$ has q 0-bits, which divide $A_{>1}$ into q segments, each starting with a 0-bit. For each j, the jth segment of the form 01^{i-1} in $A_{>1}$ represents the mapping from the jth element in $C_{>1}$ to the integer pair (a_i, b_i) in L. By Condition **C4**, at least half of the segments in $A_{>1}$ have no 1-bit, at least half of the remaining segments in $A_{>1}$ have the form 01, and at least half of the remaining segments that are not of the form 0 or 01 in $A_{>1}$ have the form 011, and so on. Therefore, the length of the binary string $A_{>1}$ is bounded by

$$\frac{q}{2} + 2\frac{q}{22} + 3\frac{q}{23} + \cdots < 2q \leq \frac{k}{2}.$$

In consequence, the number of different mappings from the

list $C_{>1}$ to the list L satisfying condition **C4** is bounded by $2^{k/2} = 1.4143^k$.

Finally, for each $c_j > 1$, the c_j-coloring F_{c_j} in the list satisfying condition **C5** is from the c_j-color coding scheme $\mathcal{F}_{c_j}$ given in Lemma 4.4, whose size is $\tau_0(c_j(c_j-1), c_j)$. The total number of lists of colorings satisfying condition **C5** is bounded by $\prod_{c_j \geq 2} \tau_0(c_j(c_j-1), c_j)$. By condition **C2**, the numbers c_j satisfy $\sum_{j=0}^{k'} c_j = k$ and $\sum_{j=0}^{k'} c_j(c_j-1) \leq 4k$. By Lemma 4.4,

$$\prod_{c_j \geq 2} \tau_0(c_j(c_j-1), c_j) \leq 2.4142^k.$$

Combining all these results, we conclude that the total number of combinations of the parameters satisfying conditions **C1**-**C5** is bounded by

$$\begin{aligned} & O(k^4) \cdot 1.8692^k \cdot O(k^{4\log k - 8}) \cdot 1.4143^k \cdot 2.4142^k \\ =& O(6.383^k k^{\log k - 4}). \end{aligned}$$

Moreover, from the above discussion, these combinations can be enumerated systematically. □

COROLLARY 5.2. *Running the algorithm* **Coloring** *in Figure 3 over all possible parameters satisfying conditions* **C1**-**C5** *gives a collection $\mathcal{F}$ of $O(6.383^k k^{\log k-4})$ k-colorings for the set Z_n.*

5.2 A new k-color coding scheme for k divisible by 4

We derive in this subsection that the collection $\mathcal{F}$ of k-colorings in Corollary 5.2 makes a k-color coding scheme for the set Z_n.

Consider the following two sets of ordered pairs of integers:

$$\begin{aligned} F_1(p) &= \{(a,b) \mid 0 < a \leq p-1 \text{ and } 0 \leq b \leq p-1\}, \\ F_2(p) &= \{(r,q) \mid 0 \leq r,q \leq p-1 \text{ and } r \neq q\}. \end{aligned}$$

Fix two distinct integers x and y, $0 \leq x, y \leq p-1$, we construct a mapping as follows:

$$\pi: \ (a,b) \longrightarrow ((ax+b) \bmod p, \ \ (ay+b) \bmod p).$$

LEMMA 5.3. *For any two integers x and y such that $0 \leq x, y \leq p-1$ and $x \neq y$, the mapping π is a one-to-one mapping from $F_1(p)$ to $F_2(p)$.*

PROOF. Since p is a prime number, the set Z_p is a field in terms of addition and multiplication modulo p. For a pair (a,b) in $F_1(p)$, from $(ax+b) \bmod p = (ay+b) \bmod p$, we would get $x = y$ (recall that p is a prime and $a \neq 0$). Therefore, the mapping π maps each element in $F_1(p)$ to an element in $F_2(p)$. Moreover, for a pair (r,q) in $F_2(p)$, where $r \neq q$, the linear equations $(ax+b) \bmod p = r$ and $(ay+b) \bmod p = q$ have a unique solution (a,b), where $a, b \in Z_p$ and $a \neq 0$, i.e., $(a,b) \in F_1(p)$. The lemma now follows directly from the fact that both sets $F_1(p)$ and $F_2(p)$ have exactly $p(p-1)$ elements. □

For a given integer s, $1 < s < n$, and an ordered pair (a,b) in $F_1(p)$, recall the function $\phi_{a,b,s}$ defined in (3) from the set Z_n to the set Z_s. For each subset W of the set Z_n and for each integer j, $0 \leq j \leq s-1$, denote by $B(a,b,s,W,j)$ the number of integers x in W such that $\phi_{a,b,s}(x) = j$. We have the following lemma.

LEMMA 5.4. *Suppose $p \bmod s = h$. Then for every subset W of k elements in Z_n, we have*

$$\begin{aligned} & \sum_{(a,b) \in F_1(p)} \sum_{j=0}^{s-1} \binom{B(a,b,s,W,j)}{2} \\ =& \frac{k(k-1)(p-h)(p-(s-h))}{2s}. \end{aligned} \tag{4}$$

PROOF. Let $p = gs + h$, where g and h are integers. Then $Z_p = \{0, 1, \ldots, gs+h-1\}$. Let

$$\begin{aligned} A_0 &= \sum_{(a,b) \in F_1(p)} \sum_{j=0}^{s-1} \binom{B(a,b,s,W,j)}{2} \\ &= \sum_{j=0}^{s-1} \sum_{(a,b) \in F_1(p)} \binom{B(a,b,s,W,j)}{2}. \end{aligned}$$

The value A_0 is equal to the number of different ways of picking an ordered pair (a,b) in $F_1(p)$, and two different elements x and y in W such that $\phi_{a,b,s}(x) = \phi_{a,b,s}(y)$, or equivalently

$$((ax+b) \bmod p) \bmod s = ((ay+b) \bmod p) \bmod s.$$

By Lemma 5.3, for two different elements x and y in W, the mapping

$$\pi: \ (a,b) \longrightarrow ((ax+b) \bmod p, \ \ (ay+b) \bmod p)$$

is a one-to-one mapping from $F_1(p)$ to $F_2(p)$. Therefore, the value A_0 is also equal to the number of different ways of picking an ordered pair (r,q) in $F_2(p)$ and two different elements x and y in W such that $r \bmod s = q \bmod s$. The number of different ways to pick two different elements x and y in W is equal to $k(k-1)/2$. Therefore, the value A_0 is equal to $k(k-1)/2$ times the number of different ways of picking an ordered pair (r,q) in $F_2(p)$ such that $r \bmod s = q \bmod s$.

For each j, if $0 \leq j \leq h-1$, then there are $g+1$ elements q in Z_p such that $q \bmod s = j$; while if $h \leq j \leq s-1$, then there are g elements q in Z_p such that $q \bmod s = j$. Therefore, for each j, $0 \leq j \leq h-1$, there are $g(g+1)$ ordered pairs (r,q) in $F_2(p)$ such that $r \bmod s = q \bmod s = j$ while for each j, $h \leq j \leq s-1$, there are $g(g-1)$ ordered pairs (r,q) in $F_2(p)$ such that $r \bmod s = q \bmod s = j$. In summary, there are totally $hg(g+1) + (s-h)g(g-1) = g(p-(s-h)) = (p-h)(p-(s-h))/s$ ordered pairs (r,q) in $F_2(p)$ such that $r \bmod s = q \bmod s$.

Therefore, we have proved

$$\begin{aligned} A_0 &= \sum_{(a,b) \in F_1(p)} \sum_{j=0}^{s-1} \binom{B(a,b,s,W,j)}{2} \\ &= \frac{k(k-1)(p-h)(p-(s-h))}{2s}. \end{aligned}$$

This completes the proof of the lemma. □

COROLLARY 5.5. *Let $1 < s < n$. For each subset W of k elements in the set Z_n, there is an ordered pair (a,b) in $F_1(p)$ such that*

$$\sum_{j=0}^{s-1} \binom{B(a,b,s,W,j)}{2} < \frac{k(k-1)}{2s}.$$

PROOF. Since there are exact $p(p-1)$ ordered pairs in $F_1(p)$, from Lemma 5.4, there is at least one ordered pair

(a,b) in $F_1(p)$ such that

$$\sum_{j=0}^{s-1}\binom{B(a,b,s,W,j)}{2} \le \frac{k(k-1)(p-h)(p-(s-h))}{2sp(p-1)}.$$

Since $p = gs+h$ is a prime number, we have $1 \le h \le s-1$ and $1 \le s-h \le s-1$. Therefore, both $(p-h)$ and $(p-(s-h))$ are not larger than $p-1$. In particular, $(p-h)(p-(s-h))$ is strictly smaller than $p(p-1)$. Now the corollary follows. □

We remark that Corollary 5.5 gives a significant improvement over the bound given by Fredman, Komlos, and Szemeredi [8], which is the bound used to implement the color coding scheme suggested by Alon, Yuster, and Zwick [1]. In particular, the bound derived in [8] uses a hash function $\psi_{a,s}$ defined by (see the proof of Theorem 3.1)

$$\psi_{a,s}(x) = (ax \bmod p) \bmod s$$

and the corresponding bound is k^2/s. On the other hand, our bound is derived based on the hash function $\phi_{a,b,s}$. The hash function $\phi_{a,b,s}$ has been studied by Carter and Wegman for other purposes [3]. Roughly speaking, the value in (4) measures the collision (i.e., the squared sum of "radii") of a hashing function. It has been shown that no hashing function can significantly improve Corollary 5.5 [3]. We will show that the bound improvement from k^2/s to $k(k-1)/(2s)$ will induce a very significant improvement on the upper bound for the size $\tau(n,k)$ of k-color coding schemes.

Now we are ready to show that the collection $\mathcal{F}$ in Corollary 5.2 makes a k-color coding scheme for the set Z_n. For this, we need to show that for every subset W of k elements in Z_n, there is a selection of the parameters satisfying conditions **C1**-**C5**, on which the algorithm **Coloring** in Figure 3 produces a k-coloring that is injective from W.

LEMMA 5.6. *For a subset W of k elements in Z_n, there is a pair (a',b') in $F_1(p)$ such that*

$$\sum_{j=0}^{k/4-1} B(a',b',k/4,W,j)(B(a',b',k/4,W,j)-1) < 4k.$$

PROOF. Let $s = k/4$. From Corollary 5.5, there is a pair (a',b') in $F_1(p)$, such that

$$\sum_{j=0}^{k/4-1}\binom{B(a',b',k/4,W,j)}{2} < 2(k-1),$$

which directly implies the lemma. □

COROLLARY 5.7. *For the subset W of k elements in Z_n, there is a pair (a',b') satisfying condition* **C1**, *such that if we let $W_j = \{x \mid \phi_{a',b',k/4}(x) = j\}$ and $c'_j = |W_j|$ for all $0 \le j \le k' = k/4-1$, then the list $C' = [c'_0,\ldots,c'_{k'}]$ satisfies condition* **C2**.

According to Corollary 5.7, there is a pair (a',b') satisfying condition **C1**, on which each set U_j constructed in step 1 of the algorithm **Coloring** contains c'_j elements in the subset W for $0 \le j \le k'$, and the list $C' = [c'_0,\ldots,c'_{k'}]$ satisfies condition **C2**.

Let $\mathcal{W}$ be a collection of some of the subsets W_j with $c'_j > 1$, as given in Corollary 5.7 ($\mathcal{W}$ may not necessarily contain all such subsets). Then we have the following lemma.

LEMMA 5.8. *Let $\mathcal{W}$ be any collection of subsets W_j of W as given in Corollary 5.7 where $c'_j = |W_j| > 1$. Then there is a pair (a'',b'') satisfying condition* **C1** *such that for at least one half of the subsets W_j in $\mathcal{W}$, the function $\phi_{a'',b'',c'_j(c'_j-1)}$ is injective from W_j to $Z_{c'_j(c'_j-1)}$.*

PROOF. Fix a subset W_j in $\mathcal{W}$, where $c'_j = |W_j| > 1$. Applying Lemma 5.4 to W_j and let $s = c'_j(c'_j-1)$ (where we have let $p \bmod s = h > 0$), we get:

$$\begin{aligned}&\sum_{(a,b)\in F_1(p)}\sum_{i=0}^{s-1}\binom{B(a,b,s,W_j,i)}{2}\\ =&\frac{c'_j(c'_j-1)(p-h)(p-(s-h))}{2s}\\ <&\frac{c'_j(c'_j-1)p(p-1)}{2s} = \frac{p(p-1)}{2}.\end{aligned}$$

Since the set $F_1(p)$ totally has $p(p-1)$ pairs, the above relation claims that for at least one half of the pairs (a,b) in $F_1(p)$, the inequality

$$\sum_{i=0}^{s-1}\binom{B(a,b,s,W_j,i)}{2} = 0$$

holds, i.e., $B(a,b,s,W_j,i) \le 1$ for all i. Therefore, for at least one half of the pairs (a,b) in $F_1(p)$, the function $\phi_{a,b,c'_j(c'_j-1)}$ is injective from the subset W_j. Applying this analysis to each subset W_j in $\mathcal{W}$ and using a simple counting argument, we derive that there is at least one pair (a'',b'') in $F_1(p)$ (i.e., a pair (a'',b'') satisfying condition **C1**) such that for at least one half of the subsets W_j in $\mathcal{W}$, the function $\phi_{a'',b'',c'_j(c'_j-1)}$ is injective from W_j. □

COROLLARY 5.9. *Let $\mathcal{W}_{>1}$ be the collection of all subsets W_j in Corollary 5.7 with $c'_j = |W_j| > 1$. Then there is a list $L' = [(a'_1,b'_1),\ldots,(a'_r,b'_r)]$ satisfying condition* **C3** *such that for at least one half of the subsets W_j in $\mathcal{W}_{>1}$, the function $\phi_{a'_1,b'_1,c'_j(c'_j-1)}$ is injective from W_j to $Z_{c'_j(c'_j-1)}$, for at least one half of the remaining subsets W_j in $\mathcal{W}_{>1}$, the function $\phi_{a'_2,b'_2,c'_j(c'_j-1)}$ is injective from W_j to $Z_{c'_j(c'_j-1)}$, and for at least one half of the remaining subsets W_j in $\mathcal{W}_{>1}$, the function $\phi_{a'_3,b'_3,c'_j(c'_j-1)}$ is injective from W_j to $Z_{c'_j(c'_j-1)}$, and so on.*

PROOF. Applying Lemma 5.8 to $\mathcal{W}_{>1}$, we get a pair (a'_1,b'_1) satisfying condition **C1** that, for at least one half of the subsets W_j in $\mathcal{W}_{>1}$, is injective from W_j. Let $\mathcal{W}'_{>1}$ be the remaining subsets in $\mathcal{W}_{>1}$. Applying Lemma 5.8 to $\mathcal{W}'_{>1}$, we get a pair (a'_2,b'_2) satisfying condition **C1** that, for at least one half of the subsets W_j in $\mathcal{W}'_{>1}$, is injective from W_j, and so on. This process stops after at most $r \le \log q$ steps, where q is the total number of subsets in $\mathcal{W}_{>1}$. Note that the list of pairs $L' = [(a'_1,b'_1),\ldots,(a'_r,b'_r)]$ constructed this way satisfies condition **C3**. □

COROLLARY 5.10. *Let W_j be the subset, $0 \le j \le k'$, as given in Corollary 5.7, $c'_j = |W_j|$, and $C' = [c'_0,\ldots,c'_{k'}]$. Then there is a list $L' = [(a'_1,b'_1),\ldots,(a'_r,b'_r)]$ satisfying condition* **C3** *and a mapping from $C'_{>1}$ to L' satisfying condition* **C4**, *such that for all j, if $c'_j > 1$ is mapped to (a'_i,b'_i), then the function $\phi_{a'_i,b'_i,c'_j(c'_j-1)}$ is injective from W_j.*

Therefore, for the pair (a',b') and the list $C' = [c'_0,\ldots,c'_{k'}]$ in Corollary 5.7, which satisfy conditions **C1** and

C2, respectively, and for the list $L' = [(a'_1, b'_1), \ldots, (a'_r, b'_r)]$ and the mapping from $C'_{>1}$ to L' in Corollary 5.10, which satisfy conditions **C3** and **C4**, respectively, each function $\phi_{a'_i,b'_i,c'_j(c'_j-1)}$ is injective from the subset W_j to $Z_{c'_j(c'_j-1)}$ for all j. For each j, let W'_j be the image of W_j under the function $\phi_{a'_i,b'_i,c'_j(c'_j-1)}$, then $W'_j \subseteq Z_{c'_j(c'_j-1)}$ and $|W'_j| = c'_j$. Now since $\mathcal{F}_{c'_j}$ is a c'_j-color coding scheme for the set $Z_{c'_j(c'_j-1)}$, one $F_{c'_j}$ of the c'_j-colorings in $\mathcal{F}_{c'_j}$ is injective from W'_j. According to the algorithm **Coloring**, when this c'_j-coloring $F_{c'_j}$ is used for the algorithm, the c'_j elements in W_j are colored with distinct colors. Running this for all j, we conclude that there is a list $[F_{c'_0}, F_{c'_1}, \ldots, F_{c'_{k'}}]$, where $F_{c'_j}$ is a c'_j-coloring in the c'_j-color coding scheme $\mathcal{F}_{c'_j}$, satisfying condition **C5** such that all elements in the subset W are colored with distinct colors.

Summarizing the above discussion, we conclude

THEOREM 5.11. *For each subset W of k elements in Z_n, there is a combination of parameters satisfying conditions* **C1**-**C5** *on which the algorithm* **Coloring** *produces a k-coloring for Z_n that is injective from W.*

Combining Theorem 5.11 with Theorem 5.1 and by running the algorithm **Coloring** on all possible combinations of parameters satisfying conditions **C1**-**C5**, we obtain a k-color coding scheme of size $O(6.383^k k^{\log k-4})$. The running time for **Coloring** is $O(k^2)$ for each colorings. Recall that we have let $n = k^2$, we get

THEOREM 5.12. *If k is divisible by 4, then $\tau(k^2, k) \leq O(6.383^k k^{\log k-4})$ and a k-color coding scheme of size $O(6.383^k k^{\log k-4})$ for the set Z_{k^2} can be constructed in time $O(6.383^k k^{\log k-2})$.*

5.3 Extension to general n and k

Using Theorem 3.1, we can easily extend Theorem 5.12 to general values of n.

THEOREM 5.13. *For any integer n, and integer k divisible by 4, $\tau(n, k) = O(6.383^k k^{\log k-4} n)$ and a k-color coding scheme of size $O(6.383^k k^{\log k-4} n)$ for the set Z_n can be constructed in time $O(6.383^k k^{\log k-2} n)$.*

To extend Theorem 5.13 to general values of k, suppose that $k = 4k' - h$, where $1 \leq h \leq 3$. We then first construct a $(4k')$-color coding scheme $\mathcal{F}'$ of size

$$O(6.383^{4k'}(4k')^{\log(4k')-4} n) = O(6.383^k k^{\log(k+h)-4} n)$$

for the set Z_n. Now for each $(4k')$-coloring F in $\mathcal{F}'$, we construct $\binom{4k'}{h} = O(k^3)$ k-colorings for Z_n by selecting every subset of h colors in F and replacing them arbitrarily by the remaining $k = 4k' - h$ colors. This gives a collection $\mathcal{F}$ of

$$O(k^3 6.383^k k^{\log(k+h)-4} n) = O(6.4^k n)$$

k-colorings for the set Z_n. To see that this is a k-color coding scheme for the set Z_n, let W be any subset of k elements in Z_n. Let W' be a subset of $4k'$ elements in Z_n obtained from W by adding arbitrarily h elements. Since $\mathcal{F}'$ is a $(4k')$-color coding scheme for Z_n, there is a $(4k')$-coloring F' in $\mathcal{F}'$ that is injective from W'. In particular, this $(4k')$-coloring F' is also injective from W. Now the k-coloring F in $\mathcal{F}$ obtained from F' by removing the other h colors is injective from W.

THEOREM 5.14. *For any integers n and k, where $n \geq k$, $\tau(n, k) = O(6.4^k n)$ and a k-color coding scheme of size $O(6.4^k n)$ for the set Z_n can be constructed in time $O(6.4^k n)$.*

6. Improved deterministic algorithms

Theorem 5.14 gives a k-color coding scheme of significantly improved size. Using this new scheme, we can immediately obtain significantly improved deterministic algorithms for PATH, MATCHING, and PACKING problems.

For the k-PATH problem, let G be a graph with n vertices. Suppose that G contains a k-path P and that we have a k-coloring on the vertices of G that is injective from the k vertices of P. Alon, Yuster, and Zwick [1] described a simple dynamic programming algorithm on such a k-colored graph that will find a k-path in the graph G in time $O(2^k km)$.

Thus, if we let $\mathcal{F}$ be the k-color coding scheme for the set Z_n in Theorem 5.14, then one of the k-colorings in $\mathcal{F}$ will be injective from the k vertices of the path P. Therefore, if we color the graph vertices using each of the k-colorings in $\mathcal{F}$, and apply the above dynamic programming algorithm on each of the k-colored graphs, then we will be able to construct a k-path in the graph G if such a path exists in the graph. This gives a deterministic algorithm of running time $O(2^k m \cdot 6.4^k n) = O(12.8^k nm)$ for the k-PATH problem, which improves the time $O(c^k n^{O(1)})$ algorithm for the problem, where $c > 8000$, based on the original k-color coding scheme in [20] suggested in [1].

Now consider the 3-D MATCHING problem. For a given set S of n triples, if a subset S_k of k disjoint triples exists, and suppose that the symbols in S are colored with $3k$ colors such that the $3k$ symbols in S_k are all colored with distinct colors. Then a similar dynamic programming process of running time $O(2^{3k} n)$ can be applied to construct a subset of k disjoint triples in S. Thus, if we use the $(3k)$-color coding scheme of size $O(6.4^{3k} n)$ in Theorem 5.14, we obtain a deterministic algorithm of running time $O(12.8^{3k} n^2)$ for the 3-D MATCHING problem. The same algorithm with very minor modifications gives an algorithm of running time $O(12.8^{3k} n^2)$ for the 3-SET PACKING problem, and an algorithm of running time $O(12.8^{3k} n^4)$ for the TRIANGLE PACKING problem.

Note that the previous best deterministic algorithms for the 3-D MATCHING and 3-SET PACKING problems [14] run in time $O(c^k n^{O(1)})$, where $c > 32000$, using the $(3k)$-color coding scheme suggested in [1]. Alternatively, Fellows et al. [7] proposed a different approach based on the color coding method. Take 3-D MATCHING as an example. Suppose that S has a subset S_k of k disjoint triples. A $(13k)$-color coding scheme of size 12.7^{3k} is constructed in [7] in which at least one $(13k)$-coloring colors all $3k$ symbols in the set S_k with distinct colors. It was left un-described in [7] how to search for the set S_k in such a $(13k)$-coloring on the symbols in the set S. Suppose the searching process takes time $O(D^{3k})$, then the algorithms proposed in [7] run in time $O((12.7D)^{3k} n^{O(1)})$ for 3-D MATCHING, 3-SET PACKING, and TRIANGLE PACKING. Using a straightforward searching process, we would have $D \geq 10.4$. On the other hand, D is at least 2 since currently the best algorithm to construct the set S_k in the set S whose symbols are colored with only $(3k)$-colors takes time $O(2^{3k} n)$. In any case, the algorithms based on our new k-color coding scheme given in Theorem 5.14 give significant improvements over the previous best deter-

ministic algorithms for these problems. In fact, combining the new color coding scheme in Theorem 5.14 with the greedy localization techniques in [4], we have been able to further improve the algorithms. In our recent work [15], we developed an $O(2.80^{3k}n^{O(1)})$ time algorithm for the 3-D MATCHING problem, and $O(4.68^{3k}n^{O(1)})$ time algorithms for the 3-SET PACKING and TRIANGLE PACKING problems. The readers are referred to Figure 1 for a more comprehensive comparison.

7. Final remarks

We have developed new randomized and deterministic algorithms for NP-hard PATH, MATCHING, and PACKING problems. Our randomized algorithms are the first group of randomized algorithms of running time $O(2^{O(k)}n^{O(1)})$ and polynomial space for these problems. Moreover, our algorithms also improve the running time of the best previous algorithms for the problems.

Our deterministic algorithms for PATH, MATCHING, and PACKING problems significantly improve the previous best algorithms for the problems. Our algorithms are based on a new k-color coding scheme of significantly improved size. A number of new techniques have been used in the development of the new k-color coding scheme, including a lower bound on the size of k-color coding schemes and a four-level hash procedure.

The color coding technique seems to provide a new approach to exact algorithms for solving NP-hard problems, in particular for those that are concerned with finding k proper elements in a set of n elements. Recent research [5] has shown evidence that for certain NP-hard problems, such searching process seems to have to take time $n^{\Omega(k)}$. Therefore, a pre-processing of k-coloring the n elements so that the k searched elements are colored distinctly seems to significantly narrow down the search space for solving the problem. For example, suppose that we have colored the vertices of a graph G with k colors such that the k vertices of a k-clique in G are all colored with distinct colors. Then searching the k-clique can be easily done in time $O((n/k)^k)$, which seems to be significantly faster than searching for the k-clique in an uncolored graph.

With more careful analysis, we can reduce the size of the k-color coding schemes to $O(6.1^k n^{O(1)})$, because of the page limit we omit the details of this improvement in this paper. And we would like to point out that it is possible to further improve this upper bound on the size of k-color coding schemes for the set Z_n. On the other hand, the lower bound 2.718^k as we derived in Theorem 3.3 has set a limit for further improvements in this direction.

8. REFERENCES

[1] N. Alon, R. Yuster, and U. Zwick, Color-coding, *Journal of the ACM* **42**, (1995), pp. 844-856.

[2] H. Bodlaender, On linear time minor tests with depth-first search, *J. Algorithms* **14**, (1993), pp. 1-23.

[3] J. L. Carter and M. N. Wegman, Universal classes of hash functions, *Journal of Computational System Science,* **18**, (1979), pp. 143-154.

[4] J. Chen, D. Friesen, I. Kanj, and W. Jia, Using nondeterminism to design efficient deterministic algorithms, *Algorithmica* **40**, (2004), pp. 83-97.

[5] J. Chen, X. Huang, I. Kanj, and G. Xia, Linear FPT reductions and computational lower bounds, *Proc. 36th ACM Symposium on Theory of Computing* (STOC 2004), pp. 212-221.

[6] R. Downey and M. Fellows, *Parameterized Complexity*, Springer-Verlag, 1999.

[7] M. Fellows, C. Knauer, N. Nishimura, P. Ragde, F. Rosamond, U. Stege, D. Thilikos, and S. Whitesides, Faster fixed-parameter tractable algorithms for matching and packing problems, *Lecture Notes in Computer Science* **3221**, (ESA 2004), pp. 311-322.

[8] M. L. Fredman, J. Komlos, and E. Szemeredi, Storing a sparse table with $O(1)$ worst case access time, *Journal of the ACM* **31**, (1984), pp. 538-544.

[9] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics*, 2nd ed., Addison-Wesley, Reading, Mass., 1994.

[10] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*, 5th ed., Oxford University Press, 1978.

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

[12] D. Karger, R. Motwani, and G. Ramkumar, On approximating the longest path in a graph, *Algorithmica* **18**, (1997), pp. 82-98.

[13] B. Kelley, R. Sharan, R. Karp, T. Sittler, D. Root, B. Stockwell, and T. Ideker, Conserved pathways within bacteria and yeast as revealed by global protein network alignment, *Proc. Natl. Acad. Sci. USA* **100**, (2003), pp. 11394-11399.

[14] I. Koutis, A faster parameterized algorithm for set packing, *Information Processing Letters* **94**, (2005), pp. 7-9.

[15] Y. Liu, S. Lu, and J. Chen, Color coding and greedy localization: improved MATCHING and SET PACKING algorithms, *Manuscript*, (2006).

[16] L. Mathieson, E. Prieto, and P. Shaw, Packing edge disjoint triangles: a parameterized view, *Lecture Notes in Computer Science* **3162** (IWPEC 2004), pp. 127-137.

[17] B. Monien, How to find long paths efficiently, *Annals of Discrete Mathematics* **25**, (1985), pp. 239-254.

[18] C. Papadimitriou and M. Yannakakis, On limited nondeterminism and the complexity of the V-C dimension, *Journal of Computer and System Sciences* **53**, (1996), pp. 161-170.

[19] E. Prieto and C. Sloper, Looking at the stars, *Theory of Computing Systems* to appear (2006).

[20] J. P. Schmidt and A. Siegel, The spatial complexity of oblivious k-probe hash functions, *SIAM Journal on Computing* **19**, (1990), pp. 775-786.

[21] J. Scott, T. Ideker, R. Karp, and R. Sharan, Efficient algorithms for detecting signaling pathways in protein interaction networks, *Lecture Notes in Bioinformatics* **3500** (RECOMB 2005), pp. 1-13.

Appendix. Proofs for Section 4

LEMMA 4.1. *For any positive integers n and k, $n \geq k$, there is a k-color coding scheme of size $\binom{n}{k}$ for the set Z_n.*

PROOF. For each subset W of k elements in Z_n, construct a k-coloring F_W that assigns each element in W a distinct color and colors all other elements in Z_n arbitrarily. The k-coloring F_W is obviously injective from W. The collection $\{F_W \mid W \subseteq Z_n,\ |W| = k\}$ of $\binom{n}{k}$ k-colorings is a k-color coding scheme for the set Z_n. □

LEMMA 4.2. *For any $n \geq 2$, $\tau(n,0) \leq 1$, $\tau(n,1) = 1$, $\tau(n,n) = 1$, and $\tau(n,2) \leq \lceil \log n \rceil$.*

PROOF. The facts $\tau(n,1) = 1$ and $\tau(n,n) = 1$ are obvious. As for $\tau(n,0)$, it is the number of colorings needed to perfectly color the empty subset and is no larger than 1. In our later discussions we let $\tau(n,0) = 1$ to ease the calculations of our analysis. We prove $\tau(n,2) \leq \lceil \log n \rceil$ by induction on n. The inequality can be easily verified for the cases of $n \leq 4$.

Inductively, suppose that for $n \leq 2^h$, $h \geq 2$, there is a 2-color coding scheme of size $\lceil \log n \rceil$ for the set Z_n. Consider now the case where $2^h < n \leq 2^{h+1}$.

Partition the n elements in Z_n into two subsets Z and Z' such that $|Z| = 2^h$ and $|Z'| = n - 2^h \leq 2^h$. By the inductive hypothesis, there exists a 2-color coding scheme $\mathcal{F} = \{F_1, \ldots, F_h\}$ of size h for Z and a 2-color coding scheme $\mathcal{F}' = \{F'_1, \ldots, F'_{h'}\}$ of size h' for Z', where $h' \leq h$.

Construct h 2-colorings for Z_n:

$$(F_1, F'_1), (F_2, F'_2), \ldots, (F_{h'}, F'_{h'}), (F_{h'+1}, F'_{h'}), \ldots, (F_h, F'_{h'}), \tag{5}$$

where the coloring (F_i, F'_j) colors the elements in Z using the 2-coloring F_i, and the elements in Z' using the 2-coloring F'_j (we assume that $\mathcal{F}'$ and $\mathcal{F}$ use the same color set). We also construct a new 2-coloring F for Z_n that assigns 0 to all elements in Z and 1 to all elements in Z'.

Let W be any subset of two elements in Z_n. If the two elements in W are both in Z or both in Z', then since $\mathcal{F}$ and $\mathcal{F}'$ are 2-color coding schemes for Z and Z', respectively, one of the 2-colorings in (5) will assign the two elements with different colors. On the other hand, if one element of W is in Z and the other element of W is in Z', then the 2-coloring F colors the two elements of W with different colors. In conclusion, the 2-coloring F plus the h 2-colorings in (5) makes a 2-color coding scheme of size $h+1 = \lceil \log n \rceil$ for the set Z_n, i.e., $\tau(n,2) \leq \lceil \log n \rceil$. □

LEMMA 4.3. *If $n = n_1 + \cdots + n_r$, where all $n_j \geq 1$, then* $\tau(n,k) \leq$

$$\sum_{0 \leq k_1 \leq n_1, \ldots, 0 \leq k_r \leq n_r}^{k_1 + \cdots + k_r = k} \left(\frac{\tau(\#[k_j \leq 1], \#[k_j = 1])}{\binom{\#[k_j \leq 1]}{\#[k_j = 1]}} \prod_{k_j \geq 2} \tau(n_j, k_j) \right),$$

where $\#[k_j \leq 1]$ and $\#[k_j = 1]$ denote the numbers of k_j's in the list $[k_1, \ldots, k_r]$ such that $k_j \leq 1$ and $k_j = 1$, respectively.

PROOF. We partition the set Z_n into r disjoint subsets Y_1, ..., Y_r as even as possible (i.e., the difference between any two of them is at most 1), where $|Y_j| = n_j$ for all $1 \leq j \leq r$. Let L be the collection of all lists $[k_1, \ldots, k_r]$ of r integers satisfying $k_1 + \cdots + k_r = k$ and $0 \leq k_j \leq n_j$ for all j. We say that two lists $[k_1, \ldots, k_r]$ and $[k'_1, \ldots, k'_r]$ in L are *conjugate* if for every j either $k_j \geq 2$ or $k'_j \geq 2$ will imply $k_j = k'_j$. It is clear that this conjugation is an equivalence relation and it partitions the lists in L into equivalence classes. A conjugation equivalence class will be called a $(k_1, \ldots, k_r)$-class for any list $[k_1, \ldots, k_r]$ in the class. Each $(k_1, \ldots, k_r)$-class contains exactly $\binom{\#[k_j \leq 1]}{\#[k_j = 1]}$ lists in L (recall that $\#[k_j \leq 1]$ and $\#[k_j = 1]$ denote the numbers of k_j's in the list $[k_1, \ldots, k_r]$ such that $k_j \leq 1$ and $k_j = 1$, respectively), because when the values and positions for all $k_j \geq 2$ are fixed in $[k_1, \ldots, k_r]$, there are exactly $\binom{\#[k_j \leq 1]}{\#[k_j = 1]}$ ways to determine the $\#[k_j = 1]$ "1"s in the remaining $\#[k_j \leq 1]$ positions in $[k_1, \ldots, k_r]$.

Fix a $(k_1, \ldots, k_r)$-class. For each j such that $k_j \geq 2$, let $\mathcal{F}_{n_j,k_j}$ be a k_j-color coding scheme of size $\tau(n_j, k_j)$ for the set Z_{n_j}. Moreover, let $\mathcal{F}$ be a $(\#[k_j = 1])$-color coding scheme of size $\tau(\#[k_j \leq 1], \#[k_j = 1])$ for the set $Z_{\#[k_j \leq 1]}$. Let the color sets used by all these schemes be disjoint. We construct

$$\tau(\#[k_j \leq 1], \#[k_j = 1]) \prod_{k_j \geq 2} \tau(n_j, k_j)$$

k-colorings for the set Z_n: each of these k-colorings consists of a k_j-coloring from the scheme $\mathcal{F}_{n_j,k_j}$ for the set Y_j for each $k_j \geq 2$, plus a $(\#[k_j = 1])$-coloring from the scheme $\mathcal{F}$ that assigns all elements in each remaining set Y_j, where $k_j \leq 1$, with a distinct color (note that there are exactly $\#[k_j \leq 1]$ such sets).

We apply the above process to each $(k_1, \ldots, k_r)$-class, which gives a collection of

$$\sum_{0 \leq k_1 \leq n_1, \ldots, 0 \leq k_r \leq n_r}^{k_1 + \cdots + k_r = k} \left(\frac{\tau(\#[k_j \leq 1], \#[k_j = 1])}{\binom{\#[k_j \leq 1]}{\#[k_j = 1]}} \prod_{k_j \geq 2} \tau(n_j, k_j) \right)$$

k-colorings for the set Z_n (note that each $(k_1, \ldots, k_r)$-class contains exactly $\binom{\#[k_j \leq 1]}{\#[k_j = 1]}$ lists in the collection L). To complete the proof of the lemma, it remains to show that this collection makes a k-color coding scheme for the set Z_n, i.e., for any subset W of k elements in Z_n, one of the above k-colorings is injective from W.

Let W be an arbitrary subset of k elements in Z_n. Suppose that for each j, W has exactly k_j elements in the set Y_j. Note that $[k_1, \ldots, k_r]$ is a list in the collection L. For each $k_j \geq 2$, since $\mathcal{F}_{n_j,k_j}$ is a k_j-color coding scheme for Y_j, one k_j-coloring F_j in $\mathcal{F}_{n_j,k_j}$ must be injective from the k_j elements of W that are in Y_j. On the other hand, since $\mathcal{F}$ is a $(\#[k_j = 1])$-color coding scheme for the set $Z_{\#[k_j \leq 1]}$, one $(\#[k_j = 1])$-coloring F in $\mathcal{F}$ assigns each of the $\#[k_j = 1]$ sets Y_j with $k_j = 1$ a distinct color. Therefore, the combination of these k_j-colorings F_j and the $(\#[k_j = 1])$-coloring F, which is one of the k-colorings constructed above, makes a k-coloring for the set Z_n that is injective from the subset W. This completes the proof of the lemma. □

By Lemma 4.2 and Lemma 4.3, for small values of n and k, we can derive specific upper bounds $\tau_0(n,k)$ for the values $\tau(n,k)$ and construct k-color coding schemes for the set Z_n. We first did this for very small values of n and k. Then using a computer program and based on Lemma 4.3, we also did this for larger values of n and k. Some of our computation results are given in Table 1 (the last column in the table will be for later discussion).

LEMMA 4.4. *There is a collection $\{\mathcal{F}_2, \mathcal{F}_3, \ldots\}$ of color coding schemes, where $\mathcal{F}_k$ is a k-color coding scheme of size*

$\tau_0(k(k-1),k)$ *for the set* $Z_{k(k-1)}$*, such that for any list of non-negative integers* $[k_1,k_2,\ldots,k_r]$ *satisfying* $k_1+\cdots+k_r = k$ *and* $\sum_{j=1}^{r} k_j(k_j-1) \le 4k$*, we have* $\prod_{k_j\ge 2}\tau_0(k_j(k_j-1),k_j) \le 2.4142^k$.

PROOF. For $k \le 18$, we use the k-color coding schemes whose size is bounded by the third column in Table 1. For $k > 18$, we simply use the upper bound $\tau_0(k(k-1),k) = \binom{k(k-1)}{k}$ given in Lemma 4.1

Since when $k = 2$, $\tau_0(k(k-1),k) = \tau_0(2,2) = 1$, we only need to take care of the case where $k \ge 3$. Let $B_k = (\tau_0(k(k-1),k))^{4/k(k-1)}$. It is easy to verify from Table 1 that $B_k \le 2.4142$ for $k \le 18$ (see the fourth column in the table).

Now consider the case $k > 18$. By our definition, $\tau_0(k(k-1),k) = \binom{k(k-1)}{k}$ when $k > 18$. Thus, $B_k = \binom{k(k-1)}{k}^{4/k(k-1)}$. Consider $f(x) = \binom{x(x-1)}{x}^{\frac{1}{x(x-1)}}$,

$$f(x) = \left[\frac{[x(x-1)]\cdots[x(x-1)-x+1]}{x!}\right]^{\frac{1}{x(x-1)}}$$

$$\le \left[\frac{[x(x-1)-\frac{x-1}{2}]^x}{\sqrt{2\pi x}(x/e)^x}\right]^{\frac{1}{x(x-1)}} < \left(\frac{e(2x-1)(x-1)}{2x}\right)^{\frac{1}{x-1}},$$

where in the first inequality, we have used the inequalities $ab \le (\frac{a+b}{2})^2 2$ and $x! \ge \sqrt{2\pi x}(x/e)^x$.

Let $g(x) = [e(2x-1)(x-1)/(2x)]^{1/(x-1)}$. It can be verified that when $x \ge 7$, $g(x)$ is strictly decreasing. In particular, for $k \ge 19$, we have

$$B_k = (f(k))^4 < (g(k))^4 \le (g(19))^4 = 2.3599.$$

Thus, $B_k \le 2.4142$ for all k. Combining this with $\sum_{j=1}^{r} k_j(k_j-1) \le 4k$, we obtain

$$\prod_{k_j\ge 2}\tau_0(k_j(k_j-1),k_j) = \prod_{k_j\ge 2} B_{k_j}^{k_j(k_j-1)/4}$$

$$\le \prod_{k_j\ge 2} 2.4142^{k_j(k_j-1)/4} = 2.4142^{\sum_{k_j\ge 2} k_j(k_j-1)/4}$$

$$= 2.4142^{\sum_{j=1}^{r} k_j(k_j-1)/4} \le 2.4142^k.$$

□

k	$n = k^2-k$	upper bound $\tau_0(n,k)$	$B_k = (\tau_0(n,k))^{\frac{4}{n}}$
2	2	1	1.0000
3	6	3	2.0801
4	12	12	2.2895
5	20	82	2.4142
6	30	434	2.2474
7	42	2,937	2.1394
8	56	16,960	2.0050
9	72	115,251	1.9108
10	90	655,756	1.8136
11	110	4,731,907	1.7488
12	132	33,489,268	1.6906
13	156	260,723,566	1.6437
14	182	1,426,381,707	1.5893
15	210	13,008,846,025	1.5584
16	240	58,465,192,360	1.5117
17	272	676,712,910,839	1.4928
18	306	6,079,615,220,515	1.4693

Table 1: Upper bound $\tau_0(n,k)$ **for** $\tau(n,k)$