

# Reconstructing Numbers from Pairwise Function Values

Shiteng Chen<sup>1</sup>, Zhiyi Huang<sup>2</sup>, and Sampath Kannan<sup>2</sup>

<sup>1</sup> Tsinghua University, Beijing 100084, China  
cst100@gmail.com

<sup>2</sup> University of Pennsylvania, Philadelphia PA 19104, USA  
{hzhayi,kannan}@cis.upenn.edu

**Abstract.** The turnpike problem is one of the few “natural” problems that are neither known to be NP-complete nor solvable by efficient algorithms. We seek to study this problem in a more general setting. We consider the generalized problem which tries to resolve set  $A = \{a_1, a_2, \dots, a_n\}$  from pairwise function values  $\{f(a_i, a_j) | 1 \leq i, j \leq n\}$  for a given bivariate function  $f$ . We call this problem the NUMBER RECONSTRUCTION problem. Our results include efficient algorithms when  $f$  is monotone and non-trivial bounds on the number of solutions when  $f$  is the sum. We also generalize previous backtracking and algebraic algorithms for the turnpike problem such that they work for the family of anti-monotone functions and linear-decomposable functions. Finally, we propose an efficient algorithm for the string reconstruction problem, which is related to an approach to protein reconstruction.

## 1 Introduction

Given a set of  $n$  numbers, it is easy to compute their pairwise distances. The reverse problem of reconstructing all possible sets of  $n$  numbers for a given unordered set of  $\binom{n}{2}$  distances is known as the *turnpike problem*. This problem dates back to the origins of X-ray crystallography in the 1930’s [11,12,13] and later arises in restriction site mapping of DNA, where it is known as the *Partial Digest Problem*, and in the area of computational geometry [15].

Rosenblatt and Seymour [14] studied a related concept called *homometric sets*. Two sets are homometric if they provide the same unordered set of pairwise distances. They introduced a generating function technique and proved two sets are homometric if and only if their generating functions have a certain relationship. We will give more details in Section 2.

Based on the generating function technique, Skiena, Lemke and Smith [8] studied the number of different solutions for a turnpike instance. Let  $H(n)$  denote the largest number of different homometric sets of size  $n$ . They proved that there exist infinitely many  $n$  such that  $H(n) \geq n^\alpha/2$ , where  $\alpha \approx 0.8107144$ , and for any  $n$ ,  $H(n) \leq n^\beta/2$ , where  $\beta \approx 1.2324827$ . Hence, the number of solutions is bounded by polynomials of the input size. They proved that the turnpike problem in arbitrary dimension is strongly NP-complete. Skiena et al.

also proposed two non-trivial algorithms for solving the problem. The first one is a combinatorial algorithm, known as the backtracking algorithm, that solves any turnpike instance in  $\mathcal{O}(2^n n \log n)$  time. Later, Zhang [19] showed that the backtracking algorithm indeed requires exponential time in the worst case, and Skiena et al. [8] proved that the backtracking algorithm works reasonably well on average if the input is drawn from a certain distribution. The second algorithm is a pseudopolynomial algorithm based on the generating function technique and polynomial factorization. The running time of this algorithm is polynomial in the maximum distance.

While hardness of the original turnpike problem remains open, there have been many successes in the study of variants. The Double Digest Problem [7] and the Simplified Partial Digest Problem [1,2], originated from other methods for reconstructing the restriction site locations of enzymes from DNA fragments and are both known to be NP-complete. Cieliebak et al. [4,3] studied four types of error that the turnpike problem may possibly encounter in real experiments. They proved that solving the turnpike problem with any of these errors is NP-complete. Pandurangan and Ramesh [10] explored a variant known as labeled partial digest. In this variant, both ends are labeled. So in addition to the pairwise distances, we are also given the set of lengths of segments at least one of whose endpoints is one of the two ends. They proposed an efficient and robust algorithm for this problem that runs in  $\mathcal{O}(n^4)$  time and tolerates an absolute error up to  $\mathcal{O}(\min_i d_i/n)$ , where  $\{d_i\}$  are the lengths of segments.

In this paper, we consider the more general NUMBER RECONSTRUCTION problem. Suppose there are  $n$  unknown integers  $a_1, a_2, \dots, a_n$  and a bivariate function  $f$ . Given an unordered set of  $n^2$  function values  $f(a_i, a_j)$ ,  $1 \leq i, j \leq n$ , the goal is to reconstruct  $a_1, a_2, \dots, a_n$  from the given set of values. We will use  $\text{REC}_f$  to denote this problem. The turnpike problem is clearly the special case when  $f$  is the difference function. We consider the following questions for a given function  $f$ . Can we solve  $\text{REC}_f$  efficiently? Can we solve  $\text{REC}_f$  uniquely? If not, what is the upper and lower bound on the number of possible solutions for any given instance? We seek to study this problem for a large family of functions  $f$  as an intermediate step toward resolving the complexity of turnpike problem.

Since the function values  $f(a_i, a_i)$  are trivially known in the turnpike case, we are also interested in a variant where we are only given an unordered set of  $n(n-1)$  function values  $f(a_i, a_j)$ ,  $1 \leq i \neq j \leq n$ . Again, the goal is to find efficient algorithms reconstructing  $a_1, a_2, \dots, a_n$  from the given values as well as bounding the number of solutions. We will use  $\text{REC}_f^*$  to denote this problem. We call this setting the *incomplete information setting* and refer to the original setting as the *full information setting*.

*Our Contribution.* We study the case when  $f$  is the sum and more generally when  $f$  is monotone. In this case, we give efficient algorithms for both the full information setting and the incomplete information setting. Furthermore, we show non-trivial bounds on the number of solutions for the case when  $f$  is the sum in incomplete information setting. We then generalize our algorithm such that it can even solve the case when  $f$  is a multi-variate function. We also

generalize previous turnpike algorithms to more general families of functions  $f$ , namely anti-monotone or linear-decomposable functions. Finally, we resolve an open problem proposed in [6] by giving an efficient algorithm for the string reconstruction problem (known as general reconstruction in [6]) that is related to a new approach to protein reconstruction. Our algorithm relies on a reduction to the turnpike problem and polynomial factorization.

## 2 Preliminaries and Notations

### 2.1 Homometric Sets

In this paper, we abuse the notion of *homometric* and say that two sets of integers  $\{a_1, a_2, \dots, a_n\}$  and  $\{b_1, b_2, \dots, b_n\}$  are homometric if they give the same set of pairwise function values, that is,  $\{f(a_i, a_j) | 1 \leq i, j \leq n\} = \{f(b_i, b_j) | 1 \leq i, j \leq n\}$ . If two homometric sets are *distinct*, then we say they are *incongruent homometric sets*. Here the meaning of distinct varies for different functions  $f$ . For example, in the turnpike problem we say two sets are distinct if and only if they are not the same under shifting and mirroring since these two operations trivially preserve the set of pairwise differences. For the case when  $f$  is the sum function, two sets are distinct simply means the sets are not equal.

### 2.2 Generating Function Technique

We will use the following *generating functions*. Given a set of integers  $A = \{a_1, a_2, \dots, a_n\}$ , let  $P_A(x)$  denote the generating function  $\sum_{i=1}^n x^{a_i}$ . Let  $D$  denote the set of pairwise differences of  $a_1, a_2, \dots, a_n$ , that is,  $\{a_i - a_j | 1 \leq i, j \leq n\}$ . We get that  $P_D(x) = P_A(x)P_A(1/x)$ . Suppose we now consider  $f$  to be the sum. Let  $S$  denote the set of pairwise sums of  $a_1, a_2, \dots, a_n$ . We get that  $P_S(x) = P_A^2(x)$ . Rosenblatt and Seymour [14] proved that two sets  $A, B$  are homometric with respect to the turnpike problem if and only if there exists two polynomials  $Q, R$  and integer  $u$  such that  $P_A(x) = Q(x)R(x)$  and  $P_B(x) = x^u Q(x)R(1/x)$ .

In this paper, we will use the generating function technique to study the number of solutions when  $f$  is the sum function. We also show that the generating function technique and polynomial factorization indeed capture the structure of NUMBER RECONSTRUCTION for a large family of functions  $f$ .

### 2.3 Measures of Polynomials

We will use the following measures of polynomials. Suppose  $P(x) = c_0 + c_1x + \dots + c_dx^d$  is a polynomial whose roots are  $\alpha_1, \alpha_2, \dots, \alpha_n$ . The Mahler Measure [18] of this polynomial  $M(P)$  is  $M(P) = c_d \prod_{i=1}^m \max\{1, |\alpha_i|\}$ . The  $L_2$  norm of the polynomial  $P$  is  $L_2(P) = (\sum_{i=0}^d c_i^2)^{1/2}$ . We have that  $M(P_1P_2) = M(P_1)M(P_2)$  and  $M(P) \leq L_2(P)$  for any polynomial  $P$ .

### 2.4 Some Notation

With a little abuse of notation, we use  $f(A)$  to denote the set  $\{f(a_i, a_j) | 1 \leq i, j \leq n\}$  for a given bivariate function  $f$  and a set  $A = \{a_1, a_2, \dots, a_n\}$ . We

shall use  $f^*(A)$  to denote the set  $\{f(a_i, a_j) | 1 \leq i \neq j \leq n\}$ . The notations can be naturally generalized to multi-variate functions  $f$ .

We let DIFF denote the difference function and let ADD denote the sum function. We will use  $H_f(n)$  to denote the maximum number of incongruent homometric sets for any instance of NUMBER RECONSTRUCTION problem with function  $f$  and  $n$  integers in the full information setting. We define  $H_f^*(n)$  similarly for the incomplete information setting.

We say a function  $f(x, y)$  is *monotone* if and only if for all  $x \geq x', y \geq y'$ ,  $f(x, y) \geq f(x', y')$ . We say a function  $f(x, y)$  is *anti-monotone* if and only if for all  $x \geq x', y \leq y'$ ,  $f(x, y) \geq f(x', y')$  (Or  $f(x, y) \leq f(x', y')$  for all  $x \geq x'$  and  $y \leq y'$ . But they are equivalent if one switches the two dimensions of  $f$ . Thus we can without loss of generality discuss only the former case). Strict monotonicity and strict anti-monotonicity can be defined straightforwardly.

We say a function  $f(x, y)$  is *linear decomposable* if there exist univariate functions  $h$  and  $g$  such that  $f(x, y) = h(x) + g(y)$ . We note that the turnpike problem and the special case of sum are both linear decomposable.

We use  $P^{\mathcal{O}}$  to denote the set of problems which can be solved in polynomial time given access to oracle  $\mathcal{O}$ .

### 3 Reconstructing Numbers from Pairwise Sums

#### 3.1 Full Information Setting

Suppose  $f$  is sum and we consider the NUMBER RECONSTRUCTION problem in the full information setting. In this setting, we can reconstruct  $a_1, a_2, \dots, a_n$  both uniquely and efficiently as illustrated in Algorithm 1.

---

#### Algorithm 1. Sum Function, Full Information

---

```

1: Sort  $S = \{a_i + a_j | 1 \leq i, j \leq n\}$ .
2: for all  $1 \leq i \leq n$  do
3:   Solve  $a_i$  from the equation  $a_1 + a_i = \max_{s \in S} s$ .
4:   if  $\{a_i + a_j | 1 \leq j \leq i\} \subseteq S$  then
5:     Let  $S = S \setminus \{a_i + a_j | 1 \leq j < i\}$ .
6:   else
7:     No solution for this instance.
8:   end if
9: end for

```

---

The key observation is that, if we assume without loss of generality that  $a_1 \geq a_2 \geq \dots \geq a_n$ , then  $a_1 + a_{i+1}$  is the largest pairwise sum in  $S$  once we have solved for  $a_1, a_2, \dots, a_i$  and removed the set  $\{a_j + a_k | 1 \leq j, k \leq i\}$  from the set  $S$  of pairwise sums, and at the first step  $a_1 + a_1$  is the largest pairwise sum. It follows from the algorithm that the solution is unique (if exists) for any given instance. The running time of the algorithm is  $\mathcal{O}(n^2)$  since steps 4-6 in the algorithm require at most  $\mathcal{O}(n)$  time.

### 3.2 Incomplete Information Setting

Now we consider reconstructing numbers from pairwise sums in the incomplete information setting. We can also find all solutions (if one exists) efficiently in this setting as in Algorithm 2. However, there may be multiple solutions for a single instance. For example,  $\{6, 3, 2, 1\}$  and  $\{5, 4, 3, 0\}$  are both valid solutions to the instance  $\{9, 8, 7, 5, 4, 3\}$ .

The correctness of the algorithm is based on the following observations. Since  $f(x, y)$  is monotone increasing in both dimensions, we have  $f(a_1, a_2) \geq f(a_1, a_3) \geq \dots \geq f(a_1, a_n)$  and  $f(a_2, a_3) \geq f(a_i, a_j)$  for any  $i, j \geq 2$  if we assume without loss of generality that  $a_1 \geq a_2 \geq \dots \geq a_n$ . So there exists some  $3 \leq k \leq n$  such that  $f(a_1, a_2) \geq f(a_1, a_3) \geq \dots \geq f(a_1, a_k) \geq f(a_2, a_3)$  are the  $k$  largest pairwise sums. Hence we can guess the correct value of  $k$  and then solve the values of  $a_1, a_2, \dots, a_k$  and finally resolve the value of  $a_{k+1}, \dots, a_n$  one by one. The running time of this algorithm is  $\mathcal{O}(n^3)$ .

---

#### Algorithm 2. Sum Function, Incomplete Information

---

```

1: Sort the set  $S = \{a_i + a_j : 1 \leq i < j \leq n, i \neq j\}$ .
2: Suppose  $S = \{s_1, s_2, \dots, s_N\}$  such that  $s_1 \geq s_2 \geq \dots \geq s_N$  and  $N = \binom{n}{2}$ .
3: for all  $3 \leq k \leq n$  do
4:   Solve  $a_1, a_2, \dots, a_k$  from equations  $a_1 + a_i = s_{i-1}$ ,  $2 \leq i \leq k$ , and  $a_2 + a_3 = s_k$ .
5:   for all  $k+1 \leq \ell \leq n$  do
6:     if  $\{a_i + a_j : 1 \leq i < j < \ell\} \subseteq S$  then
7:       Let  $S^* = S \setminus \{a_i + a_j : 1 \leq i < j < \ell\}$ .
8:       Solve  $a_\ell$  from the equation  $a_1 + a_\ell = \max_{s \in S^*} s$ .
9:     else
10:      No solution for the current value of  $k$ .
11:    end if
12:  end for
13: end for

```

---

Now let us consider the upper and lower bound on the number of solutions.

**Theorem 1.** *The following facts hold: (1) For any  $n > 2$ ,  $H_{\text{ADD}}^*(n) \leq n - 2$ . (2) For any  $n > 2$  and  $n$  is a power of 2,  $H_{\text{ADD}}^*(n) \geq 2$ .*

*Proof.* The first part of the theorem directly follows from Algorithm 2. Now we prove by induction that  $H_{\text{ADD}}^*(2^k) \geq 2$  for any  $k \geq 2$ . The base case is true because  $\{6, 3, 2, 1\}$  and  $\{5, 4, 3, 0\}$  are incongruent homometric sets. Suppose  $H_{\text{ADD}}^*(2^{k-1}) \geq 2$  for some  $k > 2$ . There exists incongruent homometric sets  $A = \{a_1, a_2, \dots, a_m\}$  and  $B = \{b_1, b_2, \dots, b_m\}$  such that  $m = |A| = |B| = 2^{k-1}$ . Let  $u$  be a large number such that  $u > |a_i - b_j|$  for any  $1 \leq i, j \leq m$ . Consider  $C = (A + u) \cup B$  and  $D = (B + u) \cup A$ , where we use  $X + u$  to denote the set  $\{x + u | x \in X\}$ . It is easy to verify that  $C$  and  $D$  are incongruent homometric sets and  $|C| = |D| = 2^k$ .  $\square$

**Theorem 2.** *The number of incongruent homometric sets  $H_{\text{ADD}}^*(n)$  is larger than 1 if and only if  $n$  is a power of 2.*

*Proof.* ( $\Leftarrow$ ) The statement is trivially true when  $n = 1, 2$ . If  $n$  is a power of 2 and  $n > 2$ , by Theorem 1 we know that  $H_{\text{ADD}}^*(n) \geq 2$ .

( $\Rightarrow$ ) Consider the following algebraic approach. Recall that given a set  $C = \{c_1, c_2, \dots, c_n\}$ ,  $P_C(x)$  is the generating function  $x^{c_1} + x^{c_2} + \dots + x^{c_n}$ . Suppose  $A = \{a_1, a_2, \dots, a_n\}$  and  $B = \{b_1, b_2, \dots, b_n\}$  are two incongruent homometric sets. We use  $S$  to denote the set of pairwise sums  $\{a_i + a_j | 1 \leq i \neq j \leq n\} = \{b_i + b_j | 1 \leq i \neq j \leq n\}$ . We have  $P_S(x) = \sum_{i \neq j} x^{a_i + a_j} = 2 \sum_{i < j} x^{a_i + a_j} = P_A(x)^2 - P_A(x^2)$  and  $P_S(x) = \sum_{i \neq j} x^{b_i + b_j} = 2 \sum_{i < j} x^{b_i + b_j} = P_B(x)^2 - P_B(x^2)$ .

So we get that  $P_A(x)^2 - P_A(x^2) = P_B(x)^2 - P_B(x^2)$ . Hence we have  $P_A(x)^2 - P_B(x)^2 = P_A(x)^2 - P_B(x)^2 = [P_A(x) - P_B(x)][P_A(x) + P_B(x)]$ . Note that  $P_A(1) = P_B(1) = n$  and hence  $P_A(1) - P_B(1) = 0$ . We have that  $(x - 1)$  divides  $P_A(x) - P_B(x)$ . Assume that  $(x - 1)^k | [P_A(x) - P_B(x)]$  and  $(x - 1)^{k+1} \nmid [P_A(x) - P_B(x)]$ , then we may assume that  $P_A(x) - P_B(x) = (x - 1)^k h(x)$  and  $h(1) \neq 1$ . We have  $(x^2 - 1)^k h(x^2) = (x - 1)^k h(x) [P_A(x) + P_B(x)]$ . Therefore  $(x + 1)^k h(x^2) = h(x) [P_A(x) + P_B(x)]$ . Note that  $P_A(1) + P_B(1) = 2n$  and  $h(1) \neq 0$ . Let  $x = 1$  and we have  $n = 2^{k-1}$ .  $\square$

## 4 The General Number Reconstruction Problem

In this section, we generalize Algorithm 1 and 2, the backtracking algorithm, and the polynomial factorization approach for more general NUMBER RECONSTRUCTION problems.

### 4.1 Monotone Functions

Note that in Algorithm 1 and 2 the key observation only rely on the fact that sum is a monotone function. We now generalize the idea in these two algorithms and propose analogous oracle algorithms for general monotone functions.

*Incomplete Information Setting.* Let us first consider the incomplete information setting. We shall discuss the case that  $f$  is symmetric and the case that  $f$  is asymmetric separately here. We also note that algorithms that resolve both cases simultaneously can be easily achieved by assuming a more powerful oracle and properly merging the algorithms we propose.

Now we study the case when  $f$  is symmetric. In this case, In order to reconstruct the numbers from the given values, we need to be able to answer some basic queries. We shall use  $\mathcal{S}_f$  to denote an oracle that can answer the following two types of queries: (1) Given  $f(x, y)$  and  $x$ , solve  $y$ ; (2) Given  $f(x, y)$ ,  $f(y, z)$ ,  $f(z, x)$ , solve  $x, y, z$ . Assuming one can resolve the above queries efficiently is reasonable because of the following fact.

**Lemma 1.** *If  $f(x, y)$  is strictly monotone, then: (1) Given  $f$  and  $x$ , there is a unique  $y$  (if exists) such that  $f(x, y) = f$ ; (2) Given  $f_1, f_2$  and  $f_3$ , there is a unique triplet  $x, y, z$  (if exists) such that  $f(x, y) = f_1, f(y, z) = f_2, f(z, x) = f_3$ .*

*Proof.* We prove both statement by contradiction. Suppose there exists two distinct  $y_1$  and  $y_2$  such that  $f(x, y_1) = f$ , and  $f(x, y_2) = f$ . Without loss of generality, we may assume that  $y_1 > y_2$ . From the strictly monotonicity we get that  $f = f(x, y_1) > f(x, y_2) = f$ , a contradiction. Now suppose there exist two distinct triples  $x_1, y_1, z_1$  and  $x_2, y_2, z_2$  such that  $f(x_1, y_1) = f(x_2, y_2) = f_1$ ,  $f(y_1, z_1) = f(y_2, z_2) = f_2$ ,  $f(z_1, x_1) = f(z_2, x_2) = f_3$ . Since two triples are distinct, without loss of generality we assume  $x_1 \neq x_2$  and thus we may further assume  $x_1 > x_2$ . If  $y_1 \geq y_2$  then from the strictly monotonicity we get  $f(x_1, y_1) > f(x_2, y_2)$ , which contradicts our assumption. So  $y_1 < y_2$ . Similarly  $z_1 < z_2$ . But now we get  $f_2 = f(y_1, z_1) < f(y_2, z_2) = f_2$ , a contradiction.  $\square$

**Theorem 3.** *Suppose  $f$  is a symmetric and monotone function, then we have  $\text{REC}_f^* \in \mathcal{P}^{\text{Sf}}$  and  $H_f^*(n) \leq n - 2$  for all  $n > 2$ .*

This theorem follows from Algorithm 3. A similar idea can be used for asymmetric monotone functions. In this case, we need strict monotonicity of function  $f$ . Again, we need to assume that some basic queries can be solved efficiently. But the second type of query is different from the symmetric case. The oracle  $\mathcal{A}_f$  answers two types of queries: (1) Given  $f(x, y)$  and  $x$ , solve  $y$ ; or given  $f(x, y)$  and  $y$ , solve  $x$ ; (2) Given  $f(x, y)$  and  $f(y, x)$ , solve  $x$  and  $y$ .

---

**Algorithm 3.** Symmetric Monotone Functions, Incomplete Information

---

```

1: Sort the set  $S = \{f(a_i, a_j) | 1 \leq i < j \leq n\}$ .
2: Suppose  $S = \{s_1, s_2, \dots, s_N\}$  such that  $s_1 \geq s_2 \geq \dots \geq s_N$  and  $N = \binom{n}{2}$ .
3: for all  $3 \leq i \leq n$  do
4:   Solve  $a_1, a_2, a_3$  given  $f(a_1, a_2) = s_1$ ,  $f(a_1, a_3) = s_2$ , and  $f(a_2, a_3) = s_i$ .
5:   for all  $4 \leq j \leq i$  do
6:     Solve  $a_j$  given  $f(a_1, a_j) = s_{j-1}$  and  $a_1$ .
7:   end for
8: for all  $i < j \leq n$  do
9:   if  $\{f(a_k, a_\ell) | 1 \leq k < \ell < j\} \subseteq S$  then
10:    Let  $S^* = S \setminus \{f(a_k, a_\ell) | 1 \leq k < \ell < j\}$ .
11:    Solve  $a_j$  given  $f(a_1, a_j) = \max_{s \in S^*} s$  and  $a_1$ .
12:   else
13:     No solution for this value  $i$ , go to next  $i$ .
14:   end if
15: end for
16: end for

```

---

One may notice that the second type of query may be unsolvable in some instances even if we assume strict monotonicity. However, if we consider the base case when  $n = 2$ , it is of exactly the same form as the second type of queries. So it is reasonable to assume one can efficiently solve at least the base case.

We shall defer the proof of the following theorem to the full version.

**Theorem 4.** *Suppose  $f$  is an asymmetric and monotone function, then we have  $\text{REC}_f^* \in \mathcal{P}^{\mathcal{A}_f}$  and  $H_f^*(n) \leq 2n - 2$  for all  $n > 2$ .*

Therefore, the NUMBER RECONSTRUCTION problem with any monotone function can be solved in polynomial time under some reasonable assumptions. Now we turn to the case with monotone functions in full information setting.

*Full Information Setting.* In the full information setting, we propose Algorithm 4 and we need an oracle  $\mathcal{F}_f$  that can answer two types of queries: (1) Given  $f(x, x)$ , solve  $x$ ; (2) Given  $f(x, y)$  and  $x$ , solve  $y$ ; or given  $f(x, y)$  and  $y$ , solve  $x$ . Again, we argue that these two types of queries are reasonable because the solution uniquely exists if we assume strict monotonicity.

**Theorem 5.** *Suppose  $f$  is an monotone function, then we have  $\text{REC}_f \in \mathcal{P}^{\mathcal{F}_f}$  and  $H_f(n) = 1$  for all  $n$ .*

*Remark 1.* We note that the above algorithms can be easily modified to solve the multi-variate version of NUMBER RECONSTRUCTION problems, in which case we want to resolve set  $A = \{a_1, a_2, \dots, a_n\}$  from the function values  $\{f(a_{i_1}, a_{i_2}, \dots, a_{i_k}) \mid 1 \leq i_1, i_2, \dots, i_k \leq n\}$ . We can define the problem for incomplete information setting similarly.

## 4.2 Anti-monotone Functions

Recall that we say a function  $f(x, y)$  is *anti-monotone* if and only if for any  $x \geq x', y \leq y', f(x, y) \geq f(x', y')$ . Thus assuming that  $a_1 \geq a_2 \geq \dots \geq a_n$ , we have the following: (1)  $f(a_1, a_n) = \max_{i,j} f(a_i, a_j)$ ; (2) Given a subset  $A' = \{a_1, \dots, a_i, a_j, \dots, a_n\} \subseteq A = \{a_1, a_2, \dots, a_n\}$  such that  $i < j - 1$ ,  $\max\{f(a_1, a_{j-1}), f(a_{i+1}, a_n)\} = \max\{f(a_k, a_\ell) \mid a_k \notin A' \vee a_\ell \notin A'\}$ . Given these two properties, we have that the backtracking algorithm solves the NUMBER RECONSTRUCTION problem for any anti-monotone functions in  $\mathcal{O}(2^n n \log n)$  time.

## 4.3 Linear Decomposable Functions

Now we consider decomposable functions  $f(x, y)$  of the form  $f(x, y) = h(x) + g(y)$ . Let  $F$  denote that set  $\{f(a_i, a_j) \mid 1 \leq i, j \leq n\}$  and let  $H$  and  $G$  denote the sets  $\{h(a_i) \mid 1 \leq i \leq n\}$  and  $\{g(a_j) \mid 1 \leq j \leq n\}$  respectively. We have that  $P_F(x) = \sum_{1 \leq i, j \leq n} x^{f(a_i, a_j)} = \sum_{1 \leq i, j \leq n} x^{h(a_i) + g(a_j)} = P_H(x)P_G(x)$ .

One can solving the NUMBER RECONSTRUCTION problem in two steps: (1) factorize a polynomial  $P_F$  of  $n^2$  terms (we count the same term multiple times if the coefficient is larger than one) into the product of two polynomials  $P'_H$  and  $P'_G$  of  $n$  terms each (2) check whether  $P_H(x) = x^u P'_H(x)$  and  $P_G(x) = x^{-u} P'_G(x)$  give a feasible solution for each  $u$ . Suppose  $P_F = x^{f_1} + x^{f_2} + \dots + x^{f_{n^2}}$  such that  $f_1 \geq f_2 \geq \dots \geq f_{n^2}$ . A naive way of factorizing is the following algorithm which capture the spirit of the backtracking algorithm:

**Algorithm 4.** Monotone Functions, Full Information

---

```

1: Sort the set  $S = \{f(a_i, a_j) | 1 \leq i, j \leq n\}$ .
2: for all  $1 \leq i \leq n$  do
3:   Solve  $a_i$  given  $f(a_1, a_i) = \max_{s \in S} s$ .
4:   if  $f(a_i, a_1) \notin S$  then
5:     Solve  $a_i$  given  $f(a_i, a_1) = \max_{s \in S} s$ .
6:   end if
7:   if  $\{f(a_i, a_i)\} \cup \{f(a_i, a_j) | 1 \leq j < i\} \cup \{f(a_j, a_i) | 1 \leq j < i\} \subseteq S$  then
8:     Let  $S^* = S \setminus (\{f(a_i, a_i)\} \cup \{f(a_i, a_j) | 1 \leq j < i\} \cup \{f(a_j, a_i) | 1 \leq j < i\})$ .
9:   else
10:    No solution for this value  $i$ , go to next  $i$ .
11:   end if
12: end for

```

---

- Without loss of generality, let  $P'_H(x) = x^{f_1}$  and  $P'_G(x) = 1$  initially.
- Guess whether  $P'_H$  or  $P'_G$  contribute the next term  $x^{f_i}$  of highest degree in  $P_F(x) - P'_H(x)P'_G(x)$ . In the first case, let  $P'_H(x) = P'_H(x) + x^{f_i}$ . In the latter case, let  $P'_G(x) = P'_G(x) + x^{f_i - f_1}$ .
- Repeat until  $P_F(x) = P'_H(x)P'_G(x)$ . If any contradiction is found (some negative term in the polynomial  $P_F(x) - P'_H(x)P'_G(x)$ ) then backtrack.

It is clear that the above algorithm is another version of backtracking algorithm. So the NUMBER RECONSTRUCTION problem for any decomposable function can be solved in  $\mathcal{O}(2^n n \log n)$  time.

The polynomial factorization approach which obtains pseudopolynomial algorithm for the turnpike problem can also be applied here. However, it is not clear that this approach still gives pseudopolynomial running time. The polynomial factorization approach proceeds in two steps: (1) factorize the polynomial  $P_F$  into the product of some irreducible polynomials and (2) for each feasible partition of these irreducible polynomials into two subsets, let  $P_H$  be the product of polynomials in one subset and let  $P_G$  be the product of polynomials in the other subset, then check if  $P_H$  and  $P_G$  give a feasible solution. Factorizing the polynomial  $P_F$  can be done in polynomial time (in  $D = \max_{1 \leq i, j \leq n} f(a_i, a_j)$ ).

*Further Discussion on the Factoring Approach.* To bound the running time of this approach we still need to bound the number of irreducible factors one need to consider when finding feasible  $P_H$  and  $P_G$ . For the turnpike problem one only need to consider non-reciprocal factors and Smyth [17] proved that  $M(P) \geq M(x^3 - x + 1) \approx 1.324$  for any non-reciprocal polynomial  $P$ . Note that  $M(P_F) \leq L_2(P_F) = n$ . We get that the number of non-reciprocal irreducible factors of  $P_F$  is bounded by  $\mathcal{O}(\log n)$ . However, we need to consider all irreducible factors for the general NUMBER RECONSTRUCTION problem. Under Lehmer's conjecture on Mahler Measure Problem [9] that  $M(P) \geq M(x^{10} + x^9 - x^7 - x^6 - x^5 - x^4 - x^3 + x + 1) \approx 1.176$ , we can bound the number of non-cyclotomic factors by  $\mathcal{O}(\log n)$ . If we can further bound the number of cyclotomic factors of  $P_F$  by  $\mathcal{O}(\log D)$ , then the factoring approach solves the NUMBER RECONSTRUCTION problem in pseudopolynomial time for linear decomposable function  $f$ .

## 5 Reconstructing Strings

Here we consider a string reconstruction problem that arises in reconstructing protein sequences [6]. Suppose there is an alphabet  $\Sigma$  and a string  $s \in \Sigma^n$ . The *profile* of a string is a vector consisting of the number of occurrences of each symbol in  $\Sigma$ . Given an unordered set of  $\binom{n+1}{2}$  profiles, one for each substring of  $s$ , the goal is to reconstruct the string  $s$ .

### 5.1 An Efficient Algorithm for Binary Alphabet

We first consider the case when the alphabet is the binary set  $B = \{\sigma_0, \sigma_1\}$ . It is clear that this problem is closely related to the turnpike problem in the following sense. If we let the symbol  $\sigma_0$  represent a segment of length 0 and let the symbol  $\sigma_1$  represent a segment of length 1. We can easily translate the given set of number of  $\sigma_0$ 's and  $\sigma_1$ 's in the substrings into the set of  $\binom{n+1}{2}$  pairwise differences of  $n+1$  integers. Moreover, the largest pairwise difference is at most  $n$ . We note that not every turnpike solution correspond to a feasible solution for the string reconstruction instance. We can verify the turnpike solutions in polynomial time since the number of solution is polynomial in input size. An alternative approach is to let  $\sigma_0$  and  $\sigma_1$  represent segments of length 1 and  $n+1$  respectively. In this approach, all turnpike solutions lead to a valid string for the original problem. Therefore, we can solve  $\text{REC}_B$  in polynomial time.

### 5.2 General Alphabet

For the string reconstruction problem with general alphabet, we can reconstruct the string bit by bit. Given a general alphabet  $\Sigma$ , let  $k = \lceil \log |\Sigma| \rceil$ . Then without loss of generality, we may assume that  $\Sigma \subseteq B^k$ . So we reconstruct the  $i^{\text{th}}$  bit of each symbol in the string using the above algorithm for binary alphabet for  $1 \leq i \leq k$ . Therefore we can solve  $\text{REC}_\Sigma$  in time polynomial in  $n$  and  $\log |\Sigma|$ . We note that Das et al. [6] proposed a different reduction from general alphabet to binary alphabet. Our reduction improves the dependence on  $|\Sigma|$ .

**Acknowledgement.** We would like to thank Alon Orlitsky for suggesting to us the string reconstruction problem.

## References

1. Abrams, Z., Chen, H.L.: The Simplified Partial Digest Problem: Hardness and a Probabilistic Analysis. In: RECOMB Satellite Meeting on DNA Sequencing Technologies and Computation (2004)
2. Blazewicz, J., Formanowicz, P., Kasprzak, M., Jaroszewski, M., Markiewicz, W.T.: Construction of DNA restriction maps based on a simplified experiment (2001)
3. Cieliebak, M., Eidenbenz, S.: Measurement errors make the partial digest problem NP-hard. In: Farach-Colton, M. (ed.) LATIN 2004. LNCS, vol. 2976, pp. 379–390. Springer, Heidelberg (2004)

4. Cieliebak, M., Eidenbenz, S., Penna, P.: Noisy data make the partial digest problem NP-hard. LNCS, pp. 111–123. Springer, Heidelberg (2003)
5. Dakic, T.: On the turnpike problem. PhD thesis, Simon Fraser University (2000)
6. Das, H., Orlicsky, A., Santhanam, N.: Order from disorder. In: Information Theory and Applications Workshop (2009)
7. Goldstein, L., Waterman, M.S.: Mapping DNA by stochastic relaxation. *Advances in Applied Mathematics* 8(2), 194–207 (1987)
8. Lemke, P., Skiena, S.S., Smith, W.D.: Reconstructing sets from inter-point distances. *Discrete and computational geometry: The Goodman-Pollack Festschrift* 25, 597–631
9. O’Byrant, K., Weisstein, E.: Lehmer’s Mahler measure problem. MathWorld—A Wolfram Web Resource
10. Pandurangan, G., Ramesh, H.: The restriction mapping problem revisited. *Journal of Computer and System Sciences* 65(3), 526–544 (2002)
11. Patterson, A.L.: A direct method for the determination of the components of interatomic distances in crystals. *Zeitschr. Krist.* 90, 517–542 (1935)
12. Patterson, A.L.: Ambiguities in the X-ray analysis of crystal structures. *Phys. Review* 65, 195–201 (1944)
13. Piccard, S.: Sur les Ensembles de Distances des Ensembles de Point d’un Espace Euclidean. *Mem. Univ. Neuchatel* 13 (1939)
14. Rosenblatt, J., Seymour, P.D.: The structure of homometric sets. *SIAM Journal on Algebraic and Discrete Methods* 3, 343 (1982)
15. Shamos, M.I.: Problems in computational geometry. Unpublished manuscript, Carnegie Mellon University, Pittsburgh, PA (1977)
16. Skiena, S.S., Sundaram, G.: A partial digest approach to restriction site mapping. *Bulletin of Mathematical Biology* 56(2), 275–294 (1994)
17. Smyth, C.J.: On the product of the conjugates outside the unit circle of an algebraic integer. *Bulletin of the London Mathematical Society* 3(2), 169 (1971)
18. Smyth, C.J.: The Mahler measure of algebraic numbers: a survey. *Number Theory and Polynomials*, 322 (2008)
19. Zhang, Z.: An exponential example for a partial digest mapping algorithm. *Journal of Computational Biology* 1(3), 235–239 (1994)