Journal of Computer and System Sciences 80 (2014) 1350–1358

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

A linear kernel for the complementary maximal strip recovery problem

Haitao Jiang [a,b], Binhai Zhu [c,*]

[a] School of Computer Science and Technology, Shandong University, Jinan 250100, China
[b] School of Mathematics, Shandong University, Jinan 250100, China
[c] Department of Computer Science, Montana State University, Bozeman, MT 59717, USA

ARTICLE INFO

Article history:
Received 7 September 2012
Received in revised form 26 January 2014
Accepted 4 March 2014
Available online 14 March 2014

Keywords:
Maximum strip recovery
FPT algorithm
Kernelization

ABSTRACT

In this paper, we compute the first linear kernel for the complementary problem of Maximal Strip Recovery (CMSR) — an NP-hard problem in computational genomics. Let k be the parameter which represents the size of the solution. The core of the technique is to first obtain a tight $18k$ bound (for the method) on the parameterized solution search space, which is done through a mixed global rules and local rules, and via an inverse amortized analysis. Then we apply additional data-reduction rules to obtain a $78k$ kernel for the problem, which is again tight for the method. Combined with the known algorithm using bounded degree search, we obtain the best Fixed-Parameterized-Tractable algorithm for CMSR to this date, running in $O(2.36^k k^2 + n^2)$ time.

1. Introduction

FPT and kernel. The rapid development of the parameterized complexity theory greatly enhances our understanding beyond NP-completeness and the traditional computational complexity theory [6,22,13]. For many theoretically intractable applications, FPT (fixed-parameter tractable) algorithms can be very effective [7,11,21].

Basically, a fixed-parameter tractable (FPT) algorithm for a *decision* problem Π with parameter k is an algorithm which solves the problem in $O(f(k)n^c) = O^*(f(k))$ time, where f is any function only on k, n is the input size and c is some fixed constant not related to k. FPT also stands for the set of problems which admit such an algorithm.

A useful technique in parameterized algorithmics is to provide polynomial time executable data-reduction rules that lead to a *problem kernel*. A data-reduction rule replaces (I,k) by an instance (I',k') in polynomial time such that: (1) $|I'| \leqslant |I|$, $k' \leqslant k$, (2) (I,k) is a Yes-instance if and only if (I',k') is a Yes-instance, and (3) $|I'| \leqslant g(k)$ for some function g. $|I'|$ is called the *size* of the kernel for the problem instance (I,k). A set of polynomial-time data-reduction rules for a problem are applied to an instance of the problem to achieve a *reduced* instance termed the *kernel*. A parameterized problem is FPT if and only if there is a polynomial time algorithm applying data-reduction rules that reduce any instance of the problem to a kernelized instance of size $g(k)$.

Kernelization is a very useful tool for designing efficient FPT algorithms [9,14]. Loosely speaking, kernelization means the reduction of the problem instance size to a function of k (k is the parameter throughout this paper). In reality, a small (especially a small linear) kernel can make it feasible to use some traditional method like branch-and-bound or ILP, so it is always meaningful. On the other hand, there are various problems which do not admit small (or even polynomial) kernels

* Corresponding author.
E-mail addresses: htjiang@sdu.edu.cn (H. Jiang), bhz@cs.montana.edu (B. Zhu).

http://dx.doi.org/10.1016/j.jcss.2014.03.005
0022-0000/

$$G_1 = \langle 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\rangle$$

$$G_2 = \langle -9, -4, -7, -6, 8, 1, 3, 2, -12, -11, -10, -5\rangle$$

$$S_1 = \langle 1, 2\rangle$$

$$S_2 = \langle 6, 7, 9\rangle$$

$$S_3 = \langle 10, 11, 12\rangle$$

$$\pi_1 = \langle 1, 2, 3\rangle$$

$$\pi_2 = \langle -2, 1, -3\rangle$$

$$G_1^\star = \langle 1, 2, 6, 7, 9, 10, 11, 12\rangle$$

$$G_2^\star = \langle -9, -7, -6, 1, 2, -12, -11, -10\rangle$$

Fig. 1. An example for the problem MSR and CMSR. MSR has a solution size of eight (with $d = 3$ strips in $G_1^\star$ and $G_2^\star$; i.e., $(1, 2)$, $(6, 7, 9)$ and $(10, 11, 12)$). CMSR has a solution size of four: the deleted markers are 3, 4, 5 and 8.

unless the polynomial hierarchy collapses to its third level [1,8,10,12]. More information about parameterized complexity can be found in the monographs [7,11,21].

In the Complementary Maximal Strip Recovery (CMSR) problem, we need to delete at most k letters from the two input sequences (signed permutations) such that the remaining letters all form into strips (or maximal common substrings of length at least two, some could be in negated and reversed form). To this date, there are two bounded search tree algorithms running in $O^*(3^k)$ [17] and $O^*(2.36^k)$ [3] respectively for CMSR, but no (linear or even polynomial) kernel is known. Part of the reason that a (linear) kernel is elusive for the CMSR is that the only known local rule (see Lemma 1, i.e., 'long' maximal common substrings can be kept as strips) is not enough to establish any polynomial kernel.

In this paper, we obtain a linear $78k$ kernel for CMSR. The core of our idea is to first bound the *parameterized* solution search space (i.e., the set of letters, whose size is a function of k, from which an optimal solution can be obtained). By applying a set of global rules (together with the local rule induced by Lemma 1), we show that this space is of size at most $18k$. On top of this we can build successfully the linear kernel of size $78k$ for CMSR.

This paper is organized as follows. In Section 2, we define the MSR and CMSR problems and the corresponding concepts for FPT formally. In Section 3, we derive the $78k$ kernel bound for CMSR. In Section 4, we close the paper with several open problems.

2. Preliminaries

MSR and CMSR. Maximal Strip Recovery (MSR) is a problem originally proposed by the David Sankoff group to eliminate noise and ambiguities in genomic maps [5,24]. In comparative genomics, a genetic map (interchangeably, a sequence) is represented by a sequence of distinct gene markers (interchangeably, letters). A gene marker can appear in two different genomic maps, in either positive or negative form. A *strip* (syntenic block) is a sequence of distinct markers that appear as subsequences in two maps, either directly or in reversed and negated form. Given two genetic maps G_1 and G_2, the problem *Maximal Strip Recovery* (MSR) [5,24] is to find two subsequences of d strips (each of length at least two), denoted as $G_i^\star$, for $i = 1, 2$, and find two signed permutations π_i of $\langle 1, \ldots, d\rangle$, such that each sequence $G_i^\star = S_{\pi_i(1)} \ldots S_{\pi_i(d)}$ (here S_{-j} denotes the reversed and negated sequence of S_j) is a subsequence of G_i, and the total length of the strips S_j is maximized. Intuitively, those gene markers not included in $G_1^\star$ and $G_2^\star$ are noise and ambiguities. The complementary problem of deleting the minimum number of noise and ambiguous markers to have a feasible solution (i.e., every remaining marker must be in some strip) is exactly the *complement of MSR*, which will be abbreviated as CMSR.

We refer to Fig. 1 for an example. In this example, each integer represents a gene marker.

Not surprisingly, in [23], both MSR and CMSR were shown to be NP-complete. Most recently, MSR was shown to be APX-hard [2,15] and CMSR was also shown to be APX-hard [16]. For positive results, in [5,24], some heuristic approaches based on MIS and Max Clique were proposed. In [4], a factor-4 polynomial-time approximation algorithm was proposed for MSR. In [17], a factor-3 polynomial-time approximation algorithm was proposed for CMSR and an $O^*(3^k)$ FPT algorithm was proposed for CMSR (the latter improves and corrects an FPT bound in [23]). Recently, the approximation factor for CMSR was improved to 2.33 [20] and the corresponding FPT algorithmic bound was improved to $O(2.36^k n^2)$ [3]. In this paper, we will focus only on the complement of MSR, or the CMSR problem.

3. A linear kernel for CMSR

Our idea for constructing the linear $78k$ kernel for CMSR is based on first identifying the *parameterized solution search space* for CMSR. Formally, a *parameterized solution search space* for the CMSR problem is a subset S of the markers in G_1, G_2 such that we only need to delete k markers in S to obtain some optimal sequences $G_1^\star$ and $G_2^\star$; moreover, $|S| \leqslant g(k)$ for some function g. Once an S (of size $18k$) is obtained, it is relatively easy to obtain the linear kernel.