Theoretical Computer Science 568 (2015) 72–83

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

# Fixed-parameter algorithms for scaffold filling ☆

Laurent Bulteau [a], Anna Paola Carrieri [b], Riccardo Dondi [c,*]

[a] Department of Software Engineering and Theoretical Computer Science, Technische Universität Berlin, Berlin, Germany
[b] Dipartimento di Informatica, Sistemistica e Comunicazione, Università degli Studi di Milano-Bicocca, Milano, Italy
[c] Dipartimento di Scienze Umane e Sociali, Università degli Studi di Bergamo, Bergamo, Italy

ARTICLE INFO

*Article history:*
Received 20 June 2014
Received in revised form 19 November 2014
Accepted 9 December 2014
Available online 16 December 2014
Communicated by G. Ausiello

*Keywords:*
Scaffold filling
Genome comparison
Computational biology
Fixed-parameter algorithms

ABSTRACT

The new sequencing technologies, called *next-generation sequencing*, provide a huge amount of data that can be used to reconstruct genomes. However, the methods applied to reconstruct genomes often are not able to reconstruct a complete genome and provide only an incomplete information. Here we consider two combinatorial problems that aim to reconstruct complete genomes by inserting a collection of missing genes. The first problem we consider, called *One-sided scaffold filling*, given an incomplete genome $B$ and a complete genome $A$, asks for the insertion of missing genes into an incomplete genome $B$ with the goal of maximizing the common adjacencies between genomes $B'$ (resulting from the insertion of missing genes in $B$) and $A$. The second problem, called *Two-sided scaffold filling*, given two incomplete genomes $A$, $B$, asks for the insertion of missing genes into both genomes so that the resulting genomes $A'$ and $B'$ have the same multiset of genes and the number of common adjacencies between $A'$ and $B'$ is maximized. Both problems were proved to be NP-hard, while their parameterized complexity, when the parameter is the number of common adjacencies of the resulting genomes, was left as an open problem. In this paper, we settle this open problem by presenting fixed-parameter algorithms for *One-sided scaffold filling* and *Two-sided scaffold filling*.



## 1. Introduction

Comparative genomics is a widely investigated field of bioinformatics in which the genomic features of different organisms are compared in order to identify biological differences and similarities. The genomic features include DNA sequences, genes, regulatory sequences and other genomic structural landmarks [22]. The ultimate goal of the approaches in this field is to understand genome functions, relationships between organisms and their evolutionary history. In this context, several interesting combinatorial problems have been introduced and studied by the computer science community (see for example [13]).

The introduction of Next Generation Sequencing (NGS) technologies leads to a huge increase on the amount of DNA/RNA and protein sequences available for genomic and trascriptomic analyses [7]. NGS technologies produce millions of short DNA/RNA fragments, called reads, that are joined together to reconstruct longer sequences. While NGS technologies generate such a huge amount of data, the cost of obtaining a complete genome is still high, in particular if compared to the cost of sequencing. Due to this fact, often released genomes are unfinished and incomplete [7]. When used in genomic analyses,

☆ A preliminary version of this paper appeared in the proceedings of ISCO 2014.

* Corresponding author.
*E-mail addresses:* l.bulteau@gmail.com (L. Bulteau), annapaola.carrieri@disco.unimib.it (A.P. Carrieri), riccardo.dondi@unibg.it (R. Dondi).

http://dx.doi.org/10.1016/j.tcs.2014.12.005
0304-3975/

incomplete draft genomes (also called *scaffolds*) may introduce errors. Hence, a relevant problem for genome comparison is the filling of scaffolds with missing genes, by means of combinatorial algorithms, in order to reconstruct complete genomes that share a high level of similarities with a known reference genome.

A combinatorial problem that has been introduced recently is the *One-sided scaffold filling problem* [20]. Such problem consists of filling a scaffold $B$ so that the resulting complete genome $B'$ minimizes the Double-Cut and Join (DCJ) distance [23] with respect to the reference genome $A$. Given two genomes, the DCJ distance is the minimum number of allowed rearrangement operations that transform one genome into the other. The authors presented a polynomial-time algorithm for the problem when the input genomes do not contain duplicated genes.

Later in [16], the scaffold filling problem has been investigated considering both the DCJ distance and the *breakpoint distance*. Given two related sequences $A$ and $B$, two consecutive elements $a_i$ and $a_{i+1}$ in $A$ form an *adjacency* if they are also consecutive in $B$ independently from the order (i.e., as $a_i a_{i+1}$ or $a_{i+1} a_i$), otherwise they form a *breakpoint*. Therefore, the breakpoint distance between $A$ and $B$ is defined as the number of breakpoints in $A$, which is equal to that of $B$. Jiang et al. [16] introduced a new related variant of the combinatorial problem, called *Two-sided scaffold filling problem*, where both genomes are incomplete. The authors show that when the input genomes do not contain gene repetitions the problem is polynomially solvable under both the DCJ distance and the breakpoint distance. However, when genomes contain duplicated genes, the scenario changes. Indeed, the authors showed that the One-sided problem is NP-complete even under the breakpoint distance.

In this paper we consider a different similarity measure to compare genomes, namely *the maximum number of common adjacencies* between two genomes. This measure has been introduced for the One-sided/Two-sided scaffold filling problems in [8]. The two problems were both proved to be NP-hard under this similarity measure [17]. The same paper has investigated the approximation complexity of the two problems, showing a 2-approximation algorithm for the Two-sided scaffold filling problem, and a $\frac{4}{3}$-approximation algorithm for the One-sided scaffold filling problem. This latter result has been recently improved in [18], where an approximation algorithm of factor $\frac{5}{4}$ for the One-sided scaffold filling problem has been presented. An approximation algorithm for a related variant of the problem has been given in [19].

In this paper, we focus on the parameterized complexity of One-sided scaffold filling and Two-sided scaffold filling. Parameterized complexity aims to investigate the computational complexity of a problem with respect to a set of interesting parameters, with the goal of understanding if the exponential explosion of an exact algorithm can eventually be confined only to the considered parameters (and not to the overall input). We refer the reader to [11,21] for an introduction to parameterized complexity.

A preliminary analysis of the parameterized complexity of the One-sided scaffold filling problem started in [17]. The authors presented two Fixed Parameter Tractable (FPT) algorithms for One-sided scaffold filling, under two different parameterizations. In the first case, they considered as parameters the number $k$ of common adjacencies between a filled genome $B'$ and a reference genome $A$, and the maximal number $d$ of occurrences of a gene inside a genome, and gave an FPT algorithm of time complexity $O((2d)^{2k} poly(|A||B|))$. In the second case, the authors considered as parameters the number $k$ of common adjacencies between a filled genome $B'$ and a reference genome $A$ and the size $c$ of the alphabet (that is the set of genes), and gave an FPT algorithm of time complexity $O(c^{2k} poly(|A||B|))$. A natural problem, left open in [17], is to consider the parameterized complexity of One-sided and Two-sided scaffold filling, when parameterized only by the maximum number of common adjacencies $k$.

**Our contribution.** In this paper we present two FTP-algorithms for both Scaffold Filling problems, thus answering the open question in [17]. More precisely, we give an algorithm of time complexity $2^{O(k)} poly(|A||B|)$ for One-sided scaffold filling and an algorithm of time complexity $2^{O(k \cdot \log k)} poly(|A||B|)$ for Two-sided scaffold filling, where $k$ is the number of common adjacencies between the resulting genomes ($A$ and $B'$ for the One-sided case, $A'$ and $B'$ for the Two-sided case). We point out that the contribution of the paper is mainly theoretical, since in practice the parameter $k$ is often close to the length of the genomes.

The rest of the paper is organized as follows. First, in Section 2 we introduce some preliminary definitions that will be useful in the rest of the paper and we give the formal definition of the two Scaffold Filling problems. In Section 3, we present the FPT algorithm for the One-sided case, while in Section 4 we present the FPT algorithm for the Two-sided case. We conclude the paper with some possible future directions.

## 2. Preliminaries

Let $\Sigma$ be an *alphabet*, that is a non-empty finite set of symbols. We represent an (unsigned) unichromosomal genome $A$ as a string over alphabet $\Sigma$. It follows that the symbols in $A$ (where each symbol represents a gene) form a multi-set on $\Sigma$, denoted by $[A]$. Consider, for example, the string $A = abcdabcdaa$ on alphabet $\Sigma = \{a, b, c, d\}$, then $[A] = \{a, a, a, a, b, b, c, c, d, d\}$. Given a string $A$, we denote by $A[i]$ the symbol of $A$ in $i$-th position, and by $A[i \dots j]$ the substring of $A$ that starts at position $i$ and ends in position $j$. Moreover, we denote the size of $A$ by $|A|$. Note that since we mostly work with multi-sets, operations $\cup$, $\cap$ and $\setminus$ are implicitely understood to be multi-set operations.

Given a string $A$, an *adjacency* of $A$ is an unordered pair of consecutive elements of $A$, that is $A[i]A[i+1]$ or $A[i+1]A[i]$, with $1 \leq i \leq |A| - 1$. We say that a position $i$, $1 \leq i \leq |A|$, *induces* an adjacency $ab$, if ($A[i] = a$ and $A[i+1] = b$) or ($A[i] = b$ and $A[i+1] = a$). We denote by $[\![A]\!]$ the multi-set of adjacencies of $A$. Following the previous example, where $A = abcdabcdaa$, we have that the multi-set of adjacencies of $A$ is $[\![A]\!] = \{aa, ab, ab, ad, ad, bc, bc, cd, cd\}$.