Journal of Discrete Algorithms 9 (2011) 12–31

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

Sparse RNA folding: Time and space efficient algorithms

Rolf Backofen [a], Dekel Tsur [b], Shay Zakov [b,*], Michal Ziv-Ukelson [b]

[a] Albert Ludwigs University, Freiburg, Germany
[b] Department of Computer Science, Ben-Gurion University of the Negev, Israel

ARTICLE INFO

Article history:
Available online 17 September 2010

Keywords:
RNA folding
Time complexity
Space complexity
Dynamic programming
Sparsification

ABSTRACT

The currently fastest algorithm for *RNA Single Strand Folding* requires $O(nZ)$ time and $\Theta(n^2)$ space, where n denotes the length of the input string and Z is a sparsity parameter satisfying $n \leqslant Z < n^2$. We show how to reduce the time and space complexities of this algorithm in the sparse case. The space reduction is based on the observation that some solutions for sub-instances are not examined after a certain stage of the algorithm, and may be discarded from memory. The running time speed up is achieved by combining two independent sparsification criteria, which restrict the number of expressions that need to be examined in bottleneck computations of the algorithm. This yields an $O(n^2 + PZ)$ time and $\Theta(Z)$ space algorithm, where P is a sparsity parameter satisfying $P < n \leqslant Z \leqslant n(P+1)$. For the base-pairing maximization variant, the time complexity is further reduced to $O(LZ)$, where L denotes the maximum number of base-pairs in a folding of the input string and satisfies $L \leqslant n\backslash 2$.
The presented techniques also extend to the related *RNA Simultaneous Alignment and Folding* problem. For an input composed of two strings of lengths n and m, the time and space complexities are reduced from $O(nm\tilde{Z})$ and $\Theta(n^2m^2)$ down to $O(n^2m^2 + \tilde{P}\tilde{Z})$ and $\Theta(nm^2 + \tilde{Z})$ respectively, where $\tilde{Z}$ and $\tilde{P}$ are sparsity parameters satisfying $\tilde{P} < nm \leqslant \tilde{Z} < nm(\tilde{P}+3)$.
A preliminary extended abstract of this work previously appeared in Backofen et al. (2009) [5]. Code implementations (in Java) may be downloaded from: http://www.cs.bgu.ac.il/~zakovs/RNAfold/SparseFold.zip.

1. Introduction

The structure of RNA is evolutionarily more conserved than its sequence and is thus key to its functional analysis [7]. Unfortunately, although massive amounts of sequence data are continuously generated, the number of known RNA structures is still relatively limited, since experimental methods, such as NMR and Crystallography, require expertise and long experimental time. Therefore, computational methods for predicting RNA structures are of significant value [40,21,39].

RNA is typically produced as a single stranded molecule, which then folds upon itself to form a number of short base-paired helices. This base-paired structure is called the *secondary structure*, or the *folding* of the RNA molecule. Secondary structures rarely contain pseudoknots (i.e. crossing base pairs). Under the assumption that the structure does not contain pseudoknots, a model was proposed by Tinoco et al. [31] to estimate the stability (in terms of free energy) of a folded RNA molecule by summing all contributions from the stabilizing, consecutive base pairs and from the loop-destabilizing terms in the secondary structure. Based on this model, dynamic programming algorithms were suggested for estimating

* Corresponding author.
E-mail addresses: backofen@informatik.uni-freiburg.de (R. Backofen), dekelts@cs.bgu.ac.il (D. Tsur), zakovs@cs.bgu.ac.il (S. Zakov), michaluz@cs.bgu.ac.il (M. Ziv-Ukelson).

1570-8667/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.jda.2010.09.001

Table 1
Time and space complexities of RNA folding algorithms. For SSF variants, n denotes the length of the input string. For SAF, n and m denote the lengths of the two input strings. $D(n)$ stands for the time complexity of computing the distance product of two $n \times n$ matrices, where the best current bound on $D(n)$ is $O(n^3 \log^3 \log n / \log^2 n)$ [8]. The sparsity parameters are bounded as follows: for SSF, $L \leqslant n/2$, $P < n \leqslant Z \leqslant n(P+1)$, and for SAF, $\tilde{P} < nm \leqslant \tilde{Z} < nm(\tilde{P}+3)$.

	Previous results		Our results	
	Time	Space	Time	Space
SSF base-pairing maximization	$\Theta(n^3)$ [26] $O(nZ)$ [34] $\Theta(D(n))$ [1]	$\Theta(n^2)$	$O(LZ)$	$\Theta(Z)$
SSF energy minimization	$\Theta(n^3)$ [41] $O(nZ)$ [34] $\Theta(D(n))$ [1]	$\Theta(n^2)$	$O(n^2 + PZ)$	$\Theta(Z)$
SAF	$\Theta(n^3m^3)$ [28] $O(nm\tilde{Z})$ [38] $\Theta(D(nm))$ [37]	$\Theta(n^2m^2)$	$O(n^2m^2 + \tilde{P}\tilde{Z})$	$\Theta(nm^2 + \tilde{Z})$

the most stable structures [33,26,41,1,34], applying various scoring criteria such as the maximal number of base pairs [26] or the minimal free energy [41]. This optimization problem is denoted here as the *RNA Single Strand Folding* (SSF) problem, and the time and space complexities of the classical algorithms for solving it are $\Theta(n^3)$ and $\Theta(n^2)$, respectively, where n denotes the length of the input RNA string. Recently, these results were sped up to yield $O(nZ)$ time and $\Theta(n^2)$ space algorithms [34], where Z is a sparsity parameter that satisfies $n \leqslant Z < n^2$. On a more theoretical front, Akutsu suggested an $\Theta(D(n))$ time and $\Theta(n^2)$ space algorithm for this problem [1], where $D(n)$ is the time for computing the distance product of two $n \times n$ matrices. The best current bound on $D(n)$ is $O(n^3 \log^3 \log n / \log^2 n)$ [8]. An additional technique which also allows to obtain a slightly sub-cubic running time was presented by Frid and Gusfield [14], resulting with an $\Theta(n^3/\log n)$ running time algorithm.

Another approach to RNA folding prediction is *RNA Simultaneous Alignment and Folding* (SAF) [28,25,18,38,35]. This approach consists of finding an optimal alignment between a set of RNA strings, where an alignment score is evaluated with respect to some common folding of the input strings. However, as stated in [16], even for the simple case where the input consists of only two strings, this approach requires "extreme amounts of memory and space" with time complexity of $\Theta(n^3m^3)$ and space complexity of $\Theta(n^2m^2)$, where n and m are the lengths of the input RNA strings to be aligned. Thus, most existing practical implementations of this algorithm [25,18,35] use restricted versions of the original problem. Since these restrictions introduce another source of error, it is of utmost practical importance to the research on RNA to improve both the space and time complexities of the full version of SAF. A first non-heuristic speedup, which does not sacrifice the optimality of results, was recently described in [38]. This work extends the approach of [34] and yields an $O(nm\tilde{Z})$ time and $\Theta(n^2m^2)$ space algorithm for the SAF problem, where $\tilde{Z}$ is a sparsity parameter that satisfies $nm \leqslant \tilde{Z} < n^2m^2$. However, experimental analysis of this algorithm indicates that the high memory requirements is a major bottleneck in practice, both in constraining the lengths of the input strings, as well as in exhausting the benchmark machine's memory, which in turn results in a page-fault slowdown (for example, for $n = m = 300$, the data structure requires about 8 Gigabyte of memory). Theoretical approaches for improving the running time of the SAF problem were recently presented [37,15]. Nevertheless, these methods are not expected to yield a significant improvement to the running time in practice, and do not improve the space complexity of the original algorithm.

Our contribution

The main contributions in this paper are as follows.

(1) *Reducing the space requirements of the SSF problem in the sparse cases*. The space requirement reduction is based on the observation that some solutions for subproblems are not examined after a certain stage of the algorithm, and may be discarded from memory. This yields an $O(nZ)$ time and $\Theta(Z)$ space algorithm for this problem (Section 3).

(2) *Reducing the time complexity of the SSF problem in the sparse cases*. We describe a faster algorithm that exploits an additional sparsity parameter P, satisfying $P < n \leqslant Z \leqslant n(P+1)$. By combining *forward dynamic programming* (previously used in [17,32,22] for related problems) with the utilization of the triangle inequality property [34], we reduce the number of sub-instance pairs that need to be considered by the algorithm and obtain an $O(n^2 + PZ)$ time and $\Theta(Z)$ space algorithm (Section 4.1). For the base-pairing maximization variant of the problem we show that $P = L \leqslant n/2$, where L denotes the maximum cardinality of a folding of the input string, and further reduce the running time to $O(LZ)$ (Section 4.2).

(3) *Extending the time and space complexity reductions to the SAF problem*. The presented sparsification techniques are adapted to the SAF problem (Section 5). The time and space complexities of the sparse algorithm are $O(n^2m^2 + \tilde{P}\tilde{Z})$ and $\Theta(nm^2 + \tilde{Z})$, respectively, where $\tilde{P} \leqslant nm \leqslant \tilde{Z} \leqslant nm(\tilde{P}+3)$.