Improved Approximation Algorithm for the Maximum Base Pair Stackings Problem in RNA Secondary Structures Prediction

Aizhong Zhou[1], Haitao Jiang[1(✉)], Jiong Guo[1], Haodi Feng[1], Nan Liu[2], and Binhai Zhu[3]

[1] School of Computer Science and Technology, Shandong University, Jinan, Shandong, China
398239146@qq.com, {htjiang,jguo,fenghaodi}@sdu.edu.cn
[2] School of Computer Science and Technology, Shandong Jianzhu University, Jinan, Shandong, China
belovedmilk@126.com
[3] Gianforte School of Computing, Montana State University, Bozeman, MT 59717-3880, USA
bhz@montana.edu

Abstract. We investigate the maximum base pair stackings problem from RNA Secondary Structures prediction in this paper. Previously, Ieong *et al.* defined a basic version of this maximum base pair stackings problem as: given an RNA sequence, finding a set of base pairs to constitute a maximum number of stackings, and proved it to be NP-hard, where the base pairs are default under some biology principle and are given implicitly. Jiang proposed a generalized version of this problem, where the candidate base pairs are given explicitly as input and presented an approximation algorithm with a factor 8/3. In this paper, we present a new approximation algorithm for the generalized maximum base pair stackings problem by a two-stage local search method, improving the approximation factor from $8/3+\varepsilon$ to 5/2. Since we adopt only two basic local operations, 1-substitutions and 2-substitutions, during the local improvement stage, the time complexity can be bounded by $O(n^7)$, much faster than the previous approximation algorithms.

1 Introduction

According to the central dogma of biology, Ribonucleic acids (RNAs) play an important role in regulating genetic and metabolic activities. Moreover, as new RNA sequences are constantly being discovered, in order to understand the biological functions of RNAs elaborately, we need to first know their structures.

An RNA is single-stranded chain and can be viewed as a sequence of nucleotides (also known as bases, denoted by A, C, G and U). The order of A, C, G, U's on the sequence form the *primary* structure of an RNA strand. An RNA folds into a three-dimensional structure by forming hydrogen bonds between nonconsecutive bases that are complementary, such as the Watson-Crick pairs

Y. Cao and J. Chen (Eds.): COCOON 2017, LNCS 10392, pp. 575–587, 2017.
DOI: 10.1007/978-3-319-62389-4_48

A-U and *C-G* and the wobble pair *G-U*. The three-dimensional arrangement of the atoms in the folded RNA molecule forms the *tertiary* structure; the collection of base pairs in the tertiary structure forms the *secondary* structure. The secondary structure can in fact tell us where there are additional connections between the bases, and where the RNA molecule could be folded. In [13], the authors claimed that "the folding of RNA is hierarchical, since secondary structure is much more stable than tertiary folding", which implies that the tertiary folding would mostly obey the secondary structure. Since the three-dimensional structure determines the function of the RNA to some extent, predicting the secondary structure of RNA becomes a key problem to study RNA in a larger and deeper scope.

In 1978, Nussinov *et al.* [9] initiated the computational study of RNA secondary structures prediction, but this problem is still not well-solved yet. The biggest impediment is the existing of *pseudoknots*, which is composed of two interleaving base pairs provided that we arrange the RNA sequence in a linear order.

In the case where there is no pseudoknot, there have been a lot of positive results. Almost all of them use a dynamic programming method [7–9,11,15, 16]. As a consequence, the optimal RNA secondary structure can be computed roughly in $O(n^3)$ time and $O(n^2)$ space.

When pseudoknots do exist in some RNAs, the secondary structures prediction problem is harder. Lyngsø and Pedersen [6] proved that determining the optimal secondary structure possibly with pseudoknots is NP-hard under some special energy functions. Akutsu [1] showed that it remains NP-hard, even if the secondary structure requires to be planar. For limited types of pseudoknots, polynomial-time algorithms have been presented [1,10,14].

According to Tinoco's energy model [12], an RNA structure can be decomposed recursively into loops with independent free energy, the stacking loops formed by two adjacent base pairs have negative energy, which stabilizes the RNA structure. Hence Ieong *et al.* [3] initiated the study for the maximum base pair stackings problem with arbitrary pseudoknots. They proved that it is NP-hard to compute the planar secondary structure with the largest number of stackings, and proposed a 2-approximation for the planar version and a 3-approximation for the general version of this problem. Later, Lyngsø [5] proved that the maximum base pair stacking loops problem without the planar restriction remains NP-hard, even for binary sequences with 0–1 base pairs. He also devised a polynomial-time approximation scheme (PTAS) for this problem, with bases over a fixed-size alphabet Σ and the base pairs being a subset of $\Sigma \times \Sigma$, which runs in $O(n^{|\Sigma|^{\frac{1}{\varepsilon}}})$ time. Unfortunately, this PTAS is impractical even for $|\Sigma| = 4$ (e.g., $\Sigma = \{A, C, G, U\}$), and $\varepsilon = 1/2$.

Among all the above results, the base pairs are given implicitly, that is, under some fix biology principle, e.g., Watson-Crick base pairs: *A-U* and *C-G*, where any two such bases can form a base pair. As an alternative, the set of candidate base pairs may be given explicitly as input, because there could be additional conditions from comparative analysis which prevents two bases from

forming a pair. This generalizes the maximum base pair stacking problem with implicit base pairs, hence the problem remains NP-hard. Jiang [4] improved the approximation factor for the maximum base pair stackings problem with explicit base pairs to $8/3+\varepsilon$. Jiang's algorithm combines the greedy strategy of Ieong's approximation algorithm and Berman's [2] approximation algorithm for computing a Maximum Weight Independent Set in $(d+1)$-claw-free graphs; to be more precise, its approximation factor is $8/3+\varepsilon$, and the time complexity is $O(n^{log_d \frac{1}{\varepsilon}})$.

In this paper, we devise a new approximation algorithm for the maximum base pair stacking problems with explicit base pairs. Our method is based on local search. The new approximation factor is 5/2, and the time complexity is $O(n^7)$.

2 Preliminaries

Let $S = s_1 s_2 \cdots s_n$ be an RNA sequence of n bases on $\{A, C, G, U\}$. We say that two bases s_i and s_{i+1} $(1 \leq i \leq n-1)$ are *continuous* on S. A secondary structure of S is a set of base pairs (s_{i_1}, s_{j_1}), (s_{i_2}, s_{j_2}), $\ldots$, (s_{i_r}, s_{j_r}), where $i_k + 2 \leq j_k$ for all $k = 1, \ldots, r$ and no two base pairs share a base. Two base pairs, such as (s_i, s_j) and (s_{i+1}, s_{j-1}) with $i + 4 \leq j$, are said to be *adjacent*. A *stacking* is a loop formed by two adjacent base pairs (s_i, s_j) and (s_{i+1}, s_{j-1}), denoted by $(s_i, s_{i+1}; s_{j-1}, s_j)$. A *helix* H of length q is composed of $q+1$ consecutive base pairs (s_i, s_j), (s_{i+1}, s_{j-1}), $\ldots$, (s_{i+q}, s_{j-q}), denoted by $(s_i, s_{i+1}, \ldots, s_{i+q}; s_{j-q}, s_{j-q+1}, \ldots, s_j)$. (s_i, s_j) and (s_{i+q}, s_{j-q}) are called *ending* base pairs of the helix H. We refer the segment of bases $s_i, s_{i+1}, \ldots, s_{i+q}$ as the α-side of the helix, and $s_{j-q}, s_{j-q+1}, \ldots, s_j$ as the β-side of the helix, denoted by H_α and H_β respectively. Note that there are exactly q stackings in a helix of length q. A helix contains at least two stackings is called a *long* helix, and a stacking is also called a *short* helix.

Now we present the formal definition of the problem to be studied in this paper. An example is shown in Fig. 1.

Problem Description: Maximum Base Pair Stackings
Input: An RNA sequence S, and a set of candidate base pairs BP.
Output: A set of chosen base pairs to constitute a maximum number of stackings.

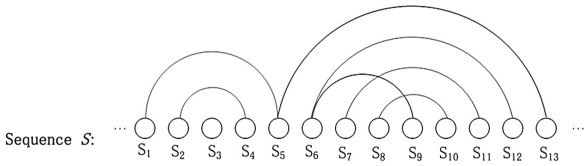

Fig. 1. The optimal base pairs found: (s_1,s_5), (s_2,s_4); and (s_6,s_{12}), (s_7,s_{11}), (s_8,s_{10}) and the maximum base pair stackings is 3. We cannot choose the base pair (s_5,s_{13}), as the base s_5 has been chosen in (s_1,s_5).

3 Our Algorithm

In this section, we depict our algorithm in detail, where the main idea of our algorithm is a two-stage local search. We firstly search for long helices by 1-substitutions and some special 2-substitutions, without considering the short helices. Then, we search for the remaining short helices, with the long helices previously found unchanged. A base is *free* if it is not involved in any stacking, otherwise, it is *occupied*. The stackings in set T are the output of our algorithm, where T is initialized as the empty set. In the algorithm, we will perform the following 3 operations of local search to obtain more stackings.

- *Operation* ① (1-substitution local improvement $\langle\ q\ \rangle$): For a helix H of length q ($q \geq 2$) in T, replace it by other long helices with a total length of $q'(q' > q)$; for a short helix in T, replace it by other short helices with a total length of $q'(q' > 1)$.
- *Operation* ② (2-substitution local improvement $\langle\ 2, p\ \rangle$ ($p \geq 3$)): For a helix H of length 2 and another helix H' of length p ($p \geq 3$) in T, replace H and an ending base pair of H' by other long helices with a total length of $p'(p' > 3)$.
- *Operation* ③ (2-substitution local improvement $\langle\ 2, 2\ \rangle$): For two helices H and H' both of length 2 in T, Replace H and H' by other long helices with a total length of $p'(p' > 4)$.

Algorithm 1. Long helices

1: **for** (q from 8 to 2) **do**
2: **while** (there exists a helix of length q, with all its bases being free) **do**
3: Put it into T; mark its bases occupied.
4: **end while**
5: **end for**
6: Perform the operation ① to the long helices in T until no other operation ① can not be performed.
7: Perform the operation ②,③ until no other operation ② and ③ can not be performed.

The following Algorithm 2 shows how to search short helices and locally improve them by 1-substitutions.

Algorithm 2. Short helices

1: **while** (there exists a short helix, with all its bases being free) **do**
2: Put it into T; mark its bases occupied.
3: **end while**
4: Perform the operation ① to the short helices in T, until no other operation ① can not be performed.

Theorem 1. *The time complexity of Algorithms 1 and 2 is* $O(n^7)$.

Proof. To generate an initial feasible solution, we search for long helices of length at most 8, it takes $O(n)$ time to find such a helix, and there are at most $O(n)$ such helices. In total, it takes $O(n^2)$ time to obtain an initial feasible solution.

During the 1-substitution local improvement process in Algorithm 1, each long helix of length q ($2 \leq q \leq 7$) occupies $2q$ bases, while we possibly make use of the 4 bases adjacent to them, so we search for long helices from these (at most) $2q + 4$ bases, which means we can obtain at most $2q + 4$ base pairs. These base pairs can constitute at most $(2q + 4)/3$ long helices. To fix a helix, it suffices to fix its starting base pair. Hence we have to search for at most $(2q+4)/3$ starting base pairs, while fixing each starting base pair takes $O(n)$ time. Consequently, for each iteration of the 1-substitution local improvement process, it takes $O(n^6)$ time.

During the 2-substitution local improvement process in Algorithm 1, we choose a helix of length 2 and a single base pair from a helix of length greater than 2, or two helices of length 2. In total they occupy 8 bases, while there are at most 8 bases adjacent to them, hence we search for long helices from these (at most) 16 bases. Similar to the above argument, it takes $O(n^5)$ time. By a similar analysis, the time complexity of the 1-substitution local improvement process in Algorithm 2 is $O(n^4)$.

Since the value of our solution is at most n, and the value of our solution would increase by at least 1 during each local improvement step, the algorithm executes at most n local improvements.

Finally, to check whether our solution cannot be further improved, we have to check all the $O(n^2)$ pairs of long helices (at least one of which must be of length 2), and check all the long helices of length less than 8 individually. In summary, the time complexity of Algorithm 1 and Algorithm 2 is $O(n^7)$. □

4 Approximation Factor Analysis

To analyze the performance of our algorithm, we should compare our solution with the optimal solution. At the termination of our algorithm, there would not be any stacking with its four bases being free, then, all the stacking in the optimal solution would either be found by our algorithm or at least one of its bases be occupied by stackings in our solution. For a stacking $T^* = (s_i, s_{i+1}; s_{j-1}, s_j)$ in the optimal solution, we say that it is *destroyed* by these helices in our solution using s_i, s_{i+1}, s_{j-1} or s_j (even if T^* is also in a stacking in our solution); moreover, it can be destroyed by H_α (or H_β), where H is helix and some bases of s_i, s_{i+1}, s_{j-1} or s_j belong to H_α (or H_β). The following lemma shows an upper bound of the number of stackings in the optimal solution which is destroyed by some helices in our solution.

Lemma 1. *A helix of length q in our solution can destroyed at most* $2q + 4$ *stackings in the optimal solution.*

Proof. Let $H = (s_i, s_{i+1}, \ldots, s_{i+q}; s_{j-q}, s_{j-q+1}, \ldots, s_j)$ be a helix of length q. It contains $q+1$ base pairs, as well as $2(q+1)$ bases. Each stacking in the optimal solution is composed of two adjacent bases in one segment. The segment of bases $s_i, s_{i+1}, \ldots, s_{i+q}$ can constitute at most $q+2$ adjacent bases together with another two bases: s_{i-1}, s_{i+q+1}. Similarly, the segment of bases $s_{j-q}, s_{j-q+1}, \ldots, s_j$ can constitute at most $q+2$ adjacent bases together with another two bases: s_{j-q-1}, s_{j+1}. In total, at most $2q+4$ stackings in the optimal solution, using these bases, could be destroyed. □

A stacking in the optimal solution is *singly* destroyed, if only one helix in our solution using its bases, otherwise, it is *multiply* destroyed.

Lemma 2. *At the termination of Algorithm 1, the number of stackings, which are singly destroyed by a long helix H of length q ($2 \leq q \leq 7$) in our solution and all of which can constitute long helices without H, is at most q.*

Proof. Since otherwise, by replacing H, we would obtain some long helices with more stackings, which means the Algorithm 1 would not terminate. □

To analyze the performance of our algorithm, we divide the stackings in the optimal solution into two parts: (1) stackings that are singly destroyed by some helices in our solution; (2) stackings that are multiply destroyed by at least two long helices in our solution. Then we assign the stackings in the optimal solution to destroyed-sets of helices by our solution. For a helix of length 8 or more in our solution, its destroyed-set DS_H contains the stackings in the optimal solution that are destroyed by it, and these stackings could appear in any other destroyed-sets. For a long helix H of length q ($2 \leq q \leq 7$) in our solution, its destroyed-set DS_H contains the stackings in the optimal solution that are destroyed by it. For a short helix H' in our solution, its destroyed-set $DS_{H'}$ contains the stackings in the optimal solution that are destroyed by it but not by any long helix. As each stacking contributes a weight of 1 to the value in the optimal solution, we assign the weight for destroyed-sets as follows:

- a singly destroyed stacking contributes a weight of 1 to the destroyed-set containing it;
- a multiply destroyed stacking contributes a weight of 1/2 to each destroyed-set containing it (see Fig. 2 for an example).

Obviously, the total weight of all the destroyed-sets would be greater than or equal to the optimal solution value, since all the stackings in the optimal solution are destroyed. To guarantee a 2.5 approximation ratio, it is sufficient to show that, given each helix H of length q, with the weight of its destroyed-set being $W(DS_H)$, it satisfies $W(DS_H)/q \leq 2.5$. Henceforth, we say a helix is *safe* if the above condition is fulfilled.

Let $H = (s_i, s_{i+1}, \ldots, s_{i+q}; s_{j-q}, s_{j-q+1}, \ldots, s_j)$ be a helix of length q in our solution, two consecutive bases s_k, s_{k+1} ($i \leq k \leq i+q-1$ or $j-q \leq k \leq j-1$) form a *gap*, if they are not in a common stacking in the optimal solution. Define $l(H_\alpha)$ and $l(H_\beta)$ to be the total length of long helices in the optimal solution,

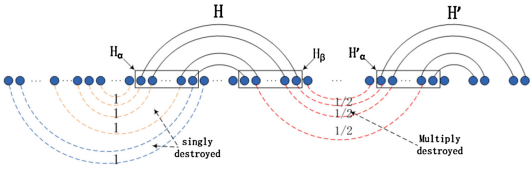

Fig. 2. The stackings in the H_α side are singly destroyed and the stackings in the other side are multiply destroyed by H and H'.

that are singly destroyed by H, and use the bases from the α-side and the β-side of H respectively. Define $s(H_\alpha)$ and $s(H_\beta)$ to be the number of short helices in the optimal solution, that are singly destroyed by H, and also use the bases from the α-side and the β-side of H respectively. Define $g(H_\alpha)$ and $g(H_\beta)$ to be the number of gaps, in the α-side and the β-side of H respectively.

Lemma 3. *At the termination of Algorithm 1, let H be a helix of length q in our solution, $s(H_\alpha) \leq g(H_\alpha) + 1 + \lfloor \frac{q+2-\max\{1,l(H_\alpha)+g(H_\alpha)\}}{5} \rfloor$, and $s(H_\beta) \leq g(H_\beta) + 1 + \lfloor \frac{q+2-\max\{1,l(H_\beta)+g(H_\beta)\}}{5} \rfloor$.*

Proof. We just prove the former inequality, since the other is exactly the same. From the definition of $s(H_\alpha)$, the short helices cannot share common base pairs, otherwise, they would form long helices. Therefore, the short helices and long helices must be spaced by gaps or multiply destroyed stackings. At the termination of Algorithm 1, our solution only contains long helices, and each long helix of length p ($p \geq 2$) can occupy $p+1$ bases, and destroy $p+2$ stackings of a helix in the optimal solution. Hence, between two short helices which are singly destroyed by H_α, if there is no gap, there should be at least four stackings which are multiply destroyed. In other words, without gaps, every 5 consecutive stacking can contain a singly destroyed short helix. In case there is no gap and no long helix, the singly destroyed short helices are spaced by segments of multiply destroyed stackings, there would be one more singly destroyed short helix. So we have an extra one in the inequality, meanwhile this short helix occupies two consecutive bases. □

In fact, it always holds that $l(H_\alpha) + s(H_\alpha) + g(H_\alpha) \leq q + 2$. Hence, when $q = 3$ and $l(H_\alpha) = 3$, we have $s(H_\alpha) \leq g(H_\alpha) \leq 1$.

Now we show that most helices are safe, except a specific case, which we would analyze separately.

Lemma 4. *A helix H of length q, where $q \geq 8$, is safe.*

Proof. From Lemma 1, H can destroy at most $2q + 4$ stackings in the optimal solution, all of which are in the destroyed-set of H. Then, we have $(2q + 4)/q \leq 2.5$, provided that $q \geq 8$. □

Lemma 5. *At the termination of Algorithm 1, a helix H of length q, where $3 \leq q \leq 7$, is safe.*

Proof. From Lemma 1, H can destroy at most $2q+4$ stackings in the optimal solution. At the termination of Algorithm 1, from Lemma 2, $l(H_\alpha)+l(H_\beta) \leq q$. Each singly destroyed stacking contributes a weight of 1 to the destroyed-set of H, and each multiply destroyed stacking contributes a weight of $1/2$ to the destroyed-set of H, totally, we can show,

$$\frac{W(DS_H)}{q} \leq \frac{3q+4+\rho}{2q} \tag{1}$$

From Lemma 3, we conclude that $\rho \leq 1$, when $q=3$; $\rho \leq 2$, when $q=4$; and $\rho \leq 3$, when $5 \leq q \leq 7$. Therefore,

$$\frac{W(DS_H)}{q} \leq 5/2 \tag{2}$$

In fact, even if we add a weight $1/2$ to the numerator, the inequality still holds. □

It remains to deal with the helices of length 2.

Lemma 6. *Let $H=(s_i, s_{i+1}, s_{i+2}; s_{j-2}, s_{j-1}, s_j)$ be a helix of length 2 in our solution, if $l(H_\alpha)$=0, then the total weight of stackings in the optimal solution assigned to H by H_α is at most 2.5.*

Proof. From Lemma 3, we have, $s(H_\alpha) \leq g(H_\alpha)+1+\lfloor \frac{2+2-\max\{1,g(H_\alpha)\}}{5} \rfloor$. That means $s(H_\alpha)-g(H_\alpha) \leq 1$. As there are at most 4 possible stackings in the optimal solution using bases of H_α, besides the gaps and short helices, all the other possible stackings can contribute a weight of $1/2$ to the destroyed-set of H, so we have $s(H_\alpha)+\frac{4-s(H_\alpha)-g(H_\alpha)}{2} \leq 2.5$. (An example is shown in Fig. 3(a)). □

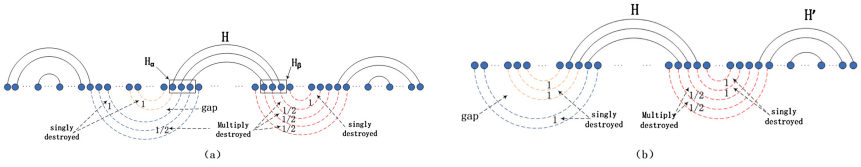

Fig. 3. (a) Case 1: the H_α side is a singly destroyed stacking with a gap, and the H_β side is a singly destroyed stacking together with three multiply destroyed stackings, and the total weight is at most be 5/2.(b) Case 2: the H_α and H_β sides decide that H gets a weight 3 when $l(H_\alpha)=2$ or $l(H_\beta)=2$.

Lemma 7. *Let $H=(s_i, s_{i+1}, s_{i+2}; s_{j-2}, s_{j-1}, s_j)$ be a helix of length 2 in our solution, if $l(H_\alpha)=2$, then the total weight of stackings in the optimal solution assigned to H by H_α is at most 3.*

Proof. From Lemma 3, we have, $s(H_\alpha) \leq g(H_\alpha)+1+\lfloor\frac{2+2-\max\{1,l(H_\alpha)+g(H_\alpha)\}}{5}\rfloor$. As there are at most 4 possible stackings in the optimal solution using bases of H_α, if $s(H_\alpha) = 1$, to split this short helix with the long helix, then $g(H_\alpha)$=1, and there would be no other possible stackings left. If $s(H_\alpha) = 0$, besides the gaps and short helices, all the other possible stackings can contribute a weight of 1/2 to the destroyed-set of H, so we have $l(H_\alpha)+s(H_\alpha)+\frac{4-l(H_\alpha)-s(H_\alpha)-g(H_\alpha)}{2} \leq 3$. (An example is shown in Fig. 3(b).) □

The following lemma can be obtained directly from Lemma 6.

Lemma 8. *Let $H = (s_i, s_{i+1}, s_{i+2}; s_{j-2}, s_{j-1}, s_j)$ be a helix of length 2 in our solution, if $l(H_\alpha) = l(H_\beta) = 0$, then H is safe.*

Obviously, there could also be helices of length 2 which is not safe. Suppose $H = (s_i, s_{i+1}, s_{i+2};\ s_{j-2}, s_{j-1}, s_j)$ is such an unsafe helix, in this case, one of $l(H_\alpha)$ and $l(H_\beta)$ could not be zero. Without loss of generality, we assume that $l(H_\alpha) = 2$ and $l(H_\beta) = 0$. From Lemma 2, we have $l(H_\alpha) + l(H_\beta) \leq 2$. Then the total weight of stackings in the optimal solution assigned to H by H_α is at most 3, and the total weight of stackings in the optimal solution assigned to H by H_β is at most 2.5. There are four possible stackings in the optimal solution using the bases of H_β, we define a weight-vector to record the weight of these four stackings contributing to H,

$$V(H_\beta) = \langle W(s_{i-1}s_i), W(s_is_{i+1}), W(s_{i+1}s_{i+2}), W(s_{i+2}s_{i+3})\rangle,$$

where $W(s_{k-1}s_k) \in \{1, 0.5, 0\}, i \leq k \leq i+3$. To make $W(s_{i-1}s_i)+W(s_is_{i+1})+W(s_{i+1}s_{i+2})+W(s_{i+2}s_{i+3}) \leq 2.5$, since there could not be two continuous 1's, $V(H_\beta)$ has two choices: either $V(H_\beta)$ contains exactly one 1 and three 0.5's; or $V(H_\beta)$ contains exactly two 1's, one 0.5, and one zero. More specifically, $V(H_\beta)$ has the following configurations: (*a*) ⟨1, 0.5, 0.5, 0.5⟩ or symmetrically ⟨0.5, 0.5, 0.5, 1⟩, (*b*) ⟨0.5, 1, 0.5, 0.5⟩ or symmetrically ⟨0.5, 0.5, 1, 0.5⟩, (*c*) ⟨1, 0, 1, 0.5⟩ or symmetrically ⟨0.5, 1, 0, 1⟩. (See Fig. 4 for examples.) We have the following lemmas.

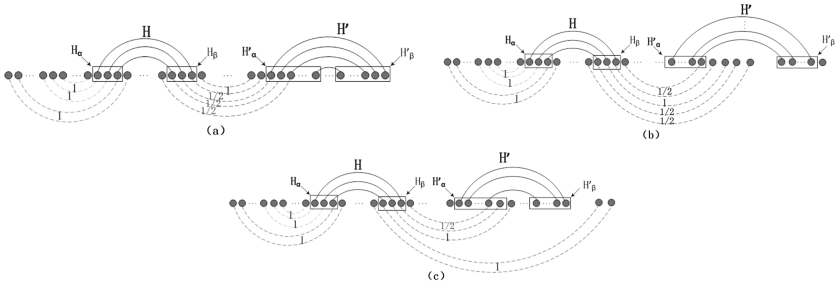

Fig. 4. (a),(b) and (c) are the examples of configuration (a),configuration (b) and configuration (c). They are all unsafe.

Lemma 9. *At the termination of Algorithm 1, configuration (a) could not exist.*

Proof. Omitted due to space constraint. □

We can observe that either configuration (b) or configuration (c) contains two continuous weight $\langle 1, 0.5\rangle$ with the weight of 1/2 being from a stacking using s_j and s_{j+1}, and the weight of 1 being from a stacking using s_{j-1} and s_j. Without loss of generality, let the stacking S^1_H using s_{j-1} and s_j be the stacking singly destroyed by H_β, and the stacking $S^{\frac{1}{2}}_H = (s_j, s_{j+1}; s_r, s_{r+1})$ be a multiply destroyed stacking, which is destroyed by both H_β and $H' = (H'_\alpha, H'_\beta)$ where $H'_\alpha = (s_{x-p}, s_{x-p+1}, \ldots, s_x)$ and $H'_\beta = (s_{y-p}, s_{y-p+1}, \ldots, s_y)$ and where $x = r$. But s_{j+1} could be possible occupied by another long helix H'' in our solution, which means the stacking $S^{\frac{1}{2}}_H$ is commonly destroyed by H, H' and H''. Then we reassign the weight of $S^{\frac{1}{2}}_H$ as follows:

- If $S^{\frac{1}{2}}_H$ is destroyed by only H and H', we assign the total weight 1 of $S^{\frac{1}{2}}_H$ to the destroyed-set of H'.
- If $S^{\frac{1}{2}}_H$ is destroyed by H, H' and H'', we assign a weight of 1/2 to each of the destroyed-set of H' and H''.

It is obviously that, under this weight assignment, H is always safe, no matter whether $V(H_\beta)$ is in configuration (b) or configuration (c). We still have to prove that under this new weight assignment, H' and H'' are safe.

Lemma 10. *At the termination of Algorithm 1, if $S^{\frac{1}{2}}_H$ is destroyed by only H_β and H'_α, then $p = 2$ and $l(H'_\alpha) = l(H'_\beta) = 0$; moreover, there cannot be a stacking in the optimal solution using s_x and s_{x-1}, which is singly destroyed by H'_α.*

Proof. We show that if any of the three consequences in the lemma does not hold, there would exist feasible local improvement for our solution, which contradicts the assumption that Algorithm 1 has terminated.

If $p \geq 3$, by removing H and the base pair (s_x, s_y), we could obtain a helix of length 2 which is singly destroyed by H, as well as a helix of length 2, which is composed of the stacking singly destroyed by H_β and an adjacent stacking multiply destroyed by both H_β and H'_α.

If $l(H'_\alpha) + l(H'_\beta) = 2$, by removing H and H', we could obtain a helix of length 2 which is singly destroyed by H, a helix of length 2 which is singly destroyed by H', as well as a helix of length 2, which is composed of the stacking singly destroyed by H_β and an adjacent stacking multiply destroyed by both H_β and H'_α.

Now consider that case when there is a stacking in the optimal solution using s_x and s_{x-1}, which is singly destroyed by H'_α. By removing H and H', we could obtain the helix of length 2 which is singly destroyed by H, as well as a helix of length 3, which is composed of the stacking singly destroyed by H_β, an adjacent stacking multiply destroyed by both H_β and H'_α, and the stacking using s_x and s_{x-1} singly destroyed by H'_α. In all these cases, a local improvement is possible. □

Lemma 11. *The weight reassignment cannot generate two new continuous stackings, both of which contributing a weight of one to the destroyed-set of H'.*

Proof. The weight assignment only involves the weight of $S_H^{\frac{1}{2}}$. If $S_H^{\frac{1}{2}}$ is destroyed by only H and H', from Lemma 10, $S_H^{\frac{1}{2}}$ cannot be adjacent to a stacking which contributes a weight of one to H'. If $S_H^{\frac{1}{2}}$ is destroyed by H, H' and H'', the weight assignment cannot generate a new stacking which contributes a weight of 1 to H' or H''. □

Lemma 11 indicates that, if H' cannot be replaced by 1-substitution after the termination of Algorithm 1, then it still cannot be replaced by any 1-substitution local improvement, even if $S_H^{\frac{1}{2}}$ is singly destroyed by H'.

Now, we show that, after the weight reassignment, H' remains safe. It can be shown similarly that H'' would be safe as well, since viewing from H, the role of H' and H'' is equivalent.

Lemma 12. *If $p \geq 3$, then H' is safe.*

Proof. From Lemma 11, no matter how much weight H' contributes from the weight reassignment, there would not be two new continuous stackings, both of which contributes a weight of one to the destroyed-set of H'. Since H' cannot be replaced by any 1-substitution local improvement, Lemma 1 still holds. Lemma 3 holds since we only keep long helix in our solution at the termination of Algorithm 1. By the same argument as in Lemma 5, we can conclude that the lemma holds. □

Lemma 13. *If $p = 2$ and $S_H^{\frac{1}{2}}$ is destroyed by only H and H', then H' is safe.*

Proof. From Lemma 11, no matter how much weight H' contributes from the weight reassignment, there would not be two continuous stackings, both of which contributes a weight of one to the destroyed-set of H'. Then, following Lemma 8, H' is safe. □

Lemma 14. *If $p = 2$ and if $S_H^{\frac{1}{2}}$ is destroyed by H, H' and H'', then H' is safe.*

Proof. Omitted due to space constraint. □

Lemma 15. *At the termination of Algorithm 2, every short helix H is safe.*

Proof. There would not be any long helix left after Algorithm 1, hence a short helix can only destroy some other short helices. A short helix occupies four bases, then it can destroy at most 4 short helices in the optimal solution. By the 1-substitution local improvement process, among the destroyed short helices, at most one of them could be singly destroyed. All the other short helices must be multiply destroyed, each of which contributing a weight of 1/2 to the destroy set of H. Hence, we have,

$$W(DS_H) \leq 1 + 3/2 = 2.5,$$

and we are done. □

Theorem 2. *Our algorithm approximates the maximum base pair stackings within a factor 5/2.*

Proof. At the termination of Algorithm 2, all the stackings in the optimal solution would be destroyed. (Recall that if some stackings are also found by our algorithm, we consider them to be destroyed by themselves.) If a stacking in the optimal solution is singly destroyed by some helix H, it contributes a weight of 1 to $W(DS_H)$; if a stacking in the optimal solution is multiply destroyed, it contributes a weight of 1/2 to two of the helices destroying it; if the weight of some stacking in the optimal solution is reassigned, its total contribution remains 1. Consequently, the total weight assigned to the destroyed-set of all the helices in our solution is exactly the value of the optimal solution. From Lemmas 12, 13, 14, 15, the weight contributed by each helix is at most 2.5 times of its own length. Since the total length of helices is exactly the value in our solution, then we are done. □

5 Concluding Remarks

In this paper, we investigate the maximum base pair stackings problem, which is a well-defined combinatorial problem from RNA secondary structures prediction. We obtain a 5/2-approximation by a two-stage local search method together with an amortization analysis. A direction for future research is to further improve the approximation factor. In our algorithm, we use only 1-substitutions and 2-substitutions. In fact, as we discussed in this paper, using 1-substitutions alone cannot reach this ratio.

Acknowledgments. This research is partially supported by NSF of China under grant 61472222 and 61202014. Haitao Jiang is supported by Young Scholars Program of Shandong University. Nan Liu is supported by he Foundation for Outstanding Young Scientists in Shandong Province (project no. BS2014DX017), by the Doctoral Foundation of Shandong Jianzhu University (project no. 0000601512). Haodi Feng is supported by NSF of China under grant 61672325. Binhai Zhu is supported by NSF of China under grant 61628207.

References

1. Akutsu, T.: Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. Discrete Appl. Mathe. **104**(1–3), 45–62 (2000)
2. Berman, P.: A $d/2$ approximation for maximum weight independent set in d-claw free graphs. Nordic J. Comput. **7**, 178–184 (2000)
3. Ieong, S., Kao, M.-Y., Lam, T.-W., Sung, W.-K., Yiu, S.-M.: Predicting RNA secondary structure with arbitrary pseudoknots by maximizing the number of stacking pairs. J. Comput. Biol. **10**, 981–995 (2003)
4. Jiang, M.: Approximation algorithms for predicting RNA secondary structures with arbitrary pseudoknots. IEEE/ACM Trans. Comput. Biol. Bioinform. **7**(2), 323–332 (2010)

5. Lyngsø, R.B.: Complexity of pseudoknot prediction in simple models. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 919–931. Springer, Heidelberg (2004). doi:10.1007/978-3-540-27836-8_77
6. Lyngsø, R.B., Pedersen, C.N.S.: RNA pseudoknot prediction in energy based models. J. Comput. Biol. **7**(3/4), 409–428 (2000)
7. Lyngsø, R.B., Zuker, M., Pedersen, C.N.S.: Fast evaluation of interval loops in rna secondary structure prediction. Bioinformatics **15**, 440–445 (1999)
8. Nussinov, R., Jacobson, A.B.: Fast algorithm for predicting the secondary structure of single-stranded RNA. Proc. Nat. Acad. Sci. USA **77**, 6309–6313 (1980)
9. Nussinov, R., Pieczenik, G., Griggs, J.R., Kleitman, D.J.: Algorithms for loop matchings. SIAM J. Appl. Mathe. **35**(1), 68–82 (1978)
10. Rivas, E., Eddy, S.R.: A dynamic programming algorithm for RNA structure prediction including pseudoknots. J. Mol. Biol. **285**(5), 2053–2068 (1999)
11. Sankoff, D.: Simultaneous solution of the RNA folding, alignment and protosequence problems. SIAM J. Appl. Mathmatics **45**, 810–825 (1985)
12. Tinoco, I., Borer, P.N., Dengler, B., Levine, M.D., Uhlenbeck, O.C., Crothers, D.M., Gralla, J.: Improved estimation of secondary structure in ribonucleic acids. Nat. New Biol. **246**, 40–42 (1973)
13. Tinoco, I., Bustamante, C.: How RNA folds. J. Mol. Biol. **293**, 271–281 (1999)
14. Uemura, Y., Hasegawa, A., Kobayashi, S., Yokomori, T.: Tree adjoining grammars for RNA structure prediction. Theore. Comput. Sci. **210**(2), 277–303 (1999)
15. Zuker, M., Sankoff, D.: RNA secondary structures and their prediction. Bull. Math. Biol. **46**, 591–621 (1984)
16. Zuker, M., Stiegler, P.: Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. Nucleic Acids Res. **9**, 133–148 (1981)