J Comb Optim (2007) 13:263–275
DOI 10.1007/s10878-006-9030-8

Improved algorithms for largest cardinality 2-interval pattern problem

Erdong Chen · Linji Yang · Hao Yuan

Published online: 9 December 2006

Abstract The 2-INTERVAL PATTERN problem is to find the largest constrained pattern in a set of 2-intervals. The constrained pattern is a subset of the given 2-intervals such that any pair of them are R-comparable, where model $R \subseteq \{<, \sqsubset, \between\}$. The problem stems from the study of general representation of RNA secondary structures. In this paper, we give three improved algorithms for different models. Firstly, an $O(n \log n + \mathcal{L})$ algorithm is proposed for the case $R = \{\between\}$, where $\mathcal{L} = O(dn) = O(n^2)$ is the total length of all 2-intervals (density d is the maximum number of 2-intervals over any point). This improves previous $O(n^2 \log n)$ algorithm. Secondly, we use dynamic programming techniques to obtain an $O(n \log n + dn)$ algorithm for the case $R = \{<, \sqsubset\}$, which improves previous $O(n^2)$ result. Finally, we present another $O(n \log n + \mathcal{L})$ algorithm for the case $R = \{\sqsubset, \between\}$ with disjoint support(interval ground set), which improves previous $O(n^2\sqrt{n})$ upper bound.

Keywords RNA secondary structure · 2-interval pattern

A preliminary version of this article appears in Proceedings of the 16th Annual International Symposium on Algorithms and Computation, Springer LNCS, Vol. 3827, pp. 412–421, Hainan, China, December 19–21, 2005.

E. Chen · L. Yang · H. Yuan (✉)
Department of Computer Science and Engineering, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai, P.R. China
e-mail: hyuan@cs.sjtu.edu.cn; hyuan1984@gmail.com

E. Chen
e-mail: edchen@cs.sjtu.edu.cn

L. Yang
e-mail: ljyang@cs.sjtu.edu.cn

1 Introduction

In the area of prediction and analysis of RNA secondary structures, arc-annotated sequence focuses on the very detailed description of the structure itself—the sequence of the bases and the bonds between the bases (Evans, 1999). However, using arc-annotated sequence to further predict other homogeneous RNA structures is proved sometimes hard. Derived from arc-annotated sequence, 2-intervals representation considers only the bonds between the bases and the patterns of the bonds, such as knots, hairpin structures and pseudoknots (Vialette, 2004). Thus, it has become a well macroscopic describer of RNA secondary structures.

A 2-interval is two disjoint intervals on a line. Two disjoint 2-intervals can be defined in the relations of precedence ($<$), nest ($\sqsubset$) or cross ($\between$). A constrained pattern is a set of 2-intervals such that any pair of them are R-comparable, where $R \subseteq \{<, \sqsubset, \between\}$. 2-INTERVAL PATTERN problem introduced by Vialette (2004) is to find the largest constrained pattern in a set of 2-intervals, and it is closely related to the problem of PATTERN MATCHING OVER SET OF 2-INTERVALS (Vialette, 2004; Gramm, 2004) and LONGEST ARC-PRESERVING COMMON SUBSEQUENCE (Evans, 1999; Alber et al., 2004; Jiang et al., 2004).

The R-comparable relations of 2-intervals can be formulated in different graph classes (Golumbic, 1980), and some graph-theoretic algorithms have been used to solve the 2-INTERVAL PATTERN problem efficiently (Vialette, 2004; Blin et al., 2004). In the paper of Blin et al. (2004), they almost completed the NP-Completeness results for 2-INTERVAL PATTERN problems under three different types of support models (unlimited, unitary, disjoint), which was classified by Vialette (2004).

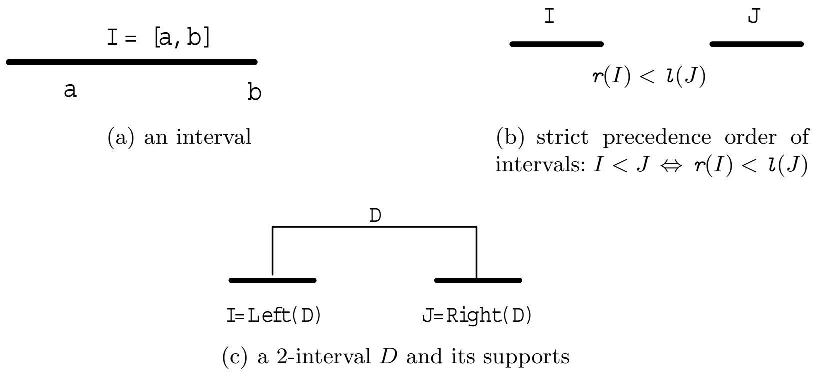
Recently, Crochemore et al. studied the approximation algorithms for 2-INTERVAL PATTERN problem (Crochemore et al., 2005). In our paper, we give several algorithms to improve the time complexity of finding optimal solutions for some models to $O(n \log n + \mathcal{L})$, which is worst-case quadratic.

The rest of this paper is organized as follows. In Section 2, we define some basic terminologies for 2-INTERVAL PATTERN problem. In Sections 3, 4 and 5, we will give improved algorithms for $R = \{\between\}$, $R = \{<, \sqsubset\}$ and $R = \{\sqsubset, \between\}$ respectively. Finally, conclusions are made in Section 6.

2 Preliminaries

First, we'll review the terminologies used in Vialette (2004) and Blin et al. (2004). Let $I = [a, b]$ be an interval ($a \leq b$), define $\mathtt{l}(I) = a$ and $\mathtt{r}(I) = b$. A 2-interval is the union of two disjoint intervals I and J, denoted by $D = (I, J)$ such that $I < J$, where the strict precedence order $<$ means I is strictly to the left of J, i.e. $\mathtt{r}(I) < \mathtt{l}(J)$. The left interval I and right interval J of D are denoted by Left(D) and Right(D) respectively (see Fig. 1).

For any two 2-intervals $D_1 = (I_1, J_1)$ and $D_2 = (I_2, J_2)$, we say they are disjoint if and only if $(I_1 \cup J_1) \cap (I_2 \cup J_2) = \emptyset$. Any pair of disjoint 2-intervals must satisfy one of the following three relations (see Fig. 2):

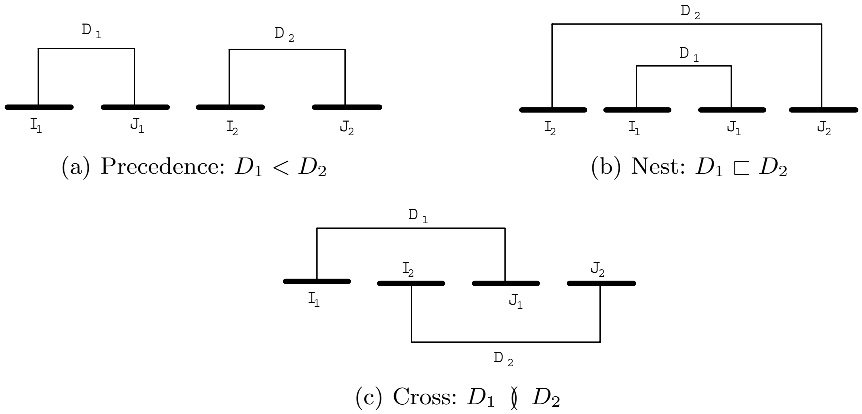


(a) an interval

(b) strict precedence order of intervals: $I < J \Leftrightarrow r(I) < l(J)$

(c) a 2-interval D and its supports

Fig. 1 Definition of 2-interval



(a) Precedence: $D_1 < D_2$

(b) Nest: $D_1 \sqsubset D_2$

(c) Cross: $D_1 \between D_2$

Fig. 2 Three relations for two disjoint 2-intervals

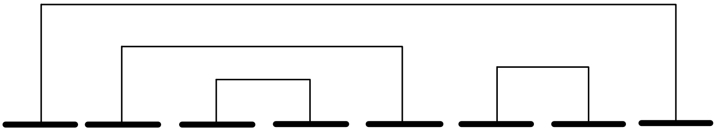


Fig. 3 Here is a $\{<, \sqsubset\}$-comparable 2-interval set, because any two 2-intervals in this set are either $\{<\}$-comparable or $\{\sqsubset\}$-comparable

PRECEDE	$D_1 < D_2$	$\Leftrightarrow$	$I_1 < J_1 < I_2 < J_2$;	transitive
NEST	$D_1 \sqsubset D_2$	$\Leftrightarrow$	$I_2 < I_1 < J_1 < J_2$;	transitive
CROSS	$D_1 \between D_2$	$\Leftrightarrow$	$I_1 < I_2 < J_1 < J_2$;	not symmetric

D_1 and D_2 are called τ-comparable if $D_1 \tau D_2$ for some $\tau \in \{<, \sqsubset, \between\}$. A 2-interval set is called R-comparable, where $R \subseteq \{<, \sqsubset, \between\}$, if and only if for any two elements of it, there exists a relation $\tau \in R$ such that they are τ-comparable (see Fig. 3 for example).

Let $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$ denote a set of n 2-intervals. The support (or interval ground set) of $\mathcal{D}$ is denoted by Support($\mathcal{D}$) $= \bigcup\{I_i, J_i \mid D_i = (I_i, J_i)\}$. Let X denote the set of interesting coordinates $\bigcup_{D \in \mathcal{D}}\{r(\text{Left}(D)), l(\text{Right}(D))\}$. Assume that the elements of $X = \{x_1, x_2, \ldots, x_{|X|}\}$ are sorted, i.e. $x_1 < x_2 < \cdots < x_{|X|}$. Since $|X| \leq 2n$, the sorting process takes $O(n \log n)$ time.

A structured pattern p is a sequence of $|P| = 2m$ words, which contains m different alphabets and each appears exactly twice. The structure pattern can be used to define a R-comparable 2-interval set. For example, $p = abaccb$ is a $\{<, \sqsubset, \between\}$-structured pattern, because $aa \between bb$, $aa < cc$ and $cc \sqsubset bb$. A R-structured pattern is called simple if $|R| = 1$. It is easy to see that $abccba$ is a simple pattern, since it is $\{\sqsubset\}$-structured.

Given a 2-interval set $\mathcal{D}$ and a model $R \subseteq \{<, \sqsubset, \between\}$, the 2-INTERVAL PATTERN problem is to find the largest cardinality subset $\mathcal{D}' \subseteq \mathcal{D}$, so that $\mathcal{D}'$ is R-comparable. In Vialette (2004), Vialette classified the problem into three types:

- UNITARY: All the intervals in Support($\mathcal{D}$) are of the same size;
- DISJOINT: The intervals in Support($\mathcal{D}$) are disjoint and equal-size;
- UNLIMITED: No restriction on the support of $\mathcal{D}$.

In our paper, we only concern the cases of DISJOINT and UNLIMITED.

To better illustrate our improvements, we define a parameter $\mathcal{L}$, which means the total length of 2-intervals. The length of a 2-interval D is defined to be Length(D) $= k_2 - k_1$, where $x_{k_1} = r(\text{Left}(D))$ and $x_{k_2} = l(\text{Right}(D))$. The density of $\mathcal{D}$, denoted by d, is the maximum number of 2-intervals over any point. Formally, $d = \max_{x \in X} |\{D = (I, J) \in \mathcal{D} \mid r(I) \leq x < l(J)\}|$. It is easy to see $\mathcal{L} \leq dn \leq n^2$. Table 1 summarizes the results of our work.

Table 1 Summarized results for different models of 2-INTERVAL PATTERN problem

	Support	
Model	Disjoint	Unlimited
$\{<, \sqsubset, \between\}$	$O(n\sqrt{n})$ (Micali and Vazirani, 1980)	APX-Hard (Bar-Yehuda et al., 2002)
$\{<, \between\}$	?	NP-Complete (Blin et al., 2004)
$\{\sqsubset, \between\}$	$O(n \log n + \mathcal{L})^*$	NP-Complete (Vialette, 2004)
$\{<, \sqsubset\}$	$O(n \log n + dn)^*$	
$\{\between\}$	$O(n \log n + \mathcal{L})^*$	
$\{<\}$	$O(n \log n)$ (Vialette, 2004)	
$\{\sqsubset\}$	$O(n \log n)$ (Blin et al., 2004)	

The improved results of this paper are marked by $*$. The symbol " ? " means the complexity of that case remains open.

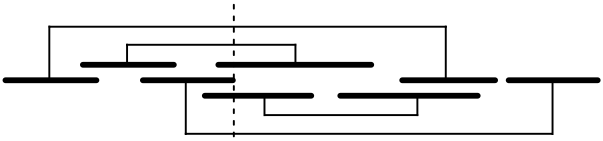


Fig. 4 A sweep-line sweeps from left to right

3 Improved algorithm for $\{\between\}$-structured pattern

In this section, we will give an $O(n\log n + \mathcal{L})$ algorithm for the model $R = \{\between\}$. This improves the $O(n^2\log n)$ algorithm given in Vialette (2004).

Our algorithm is based on a sweep-line method. Let $\mathcal{D}[x]$ be the set of 2-intervals crossing a vertical line whose horizontal coordinate is x, that is $\mathcal{D}[x] = \{D_i \in \mathcal{D} \mid r(\text{Left}(D_i)) \le x \text{ and } l(\text{Right}(D_i)) > x\}$. For convenience, set $x_0 = -\infty$ and let $\mathcal{D}^{(k)}$ denote $\mathcal{D}[x_k]$ for $0 \le k \le |X|$, hence $\mathcal{D}^{(0)} = \emptyset$. When the vertical line sweeps from left to right (Fig. 4) passingan interesting point $x_k \in X$, some 2-intervals may be added to $\mathcal{D}[x]$, and some may be removed. Let $P^{(k)}$ and $Q^{(k)}$ be the differences between $\mathcal{D}^{(k-1)}$ and $\mathcal{D}^{(k)}$, we have

$$P^{(k)} = \mathcal{D}^{(k)} \setminus \mathcal{D}^{(k-1)} = \{D \in \mathcal{D} \mid r(\text{Left}(D)) = x_k\},$$
$$Q^{(k)} = \mathcal{D}^{(k-1)} \setminus \mathcal{D}^{(k)} = \{D \in \mathcal{D} \mid l(\text{Right}(D)) = x_k\}.$$

Theorem 1. *Let $\omega(\mathcal{D})$ be the cardinality of the largest $\{\between\}$-comparable subset of $\mathcal{D}$, then $\omega(\mathcal{D}) = \max_{1\le k\le |X|}\ \omega(\mathcal{D}^{(k)})$.*

Proof: Observe that for any two 2-intervals D_i and D_j which are $\between$-comparable, we have $r(\text{Left}(D_j)) < l(\text{Right}(D_i))$. Thus for any subset $\mathcal{D}' \subseteq \mathcal{D}$ which is $\{\between\}$-comparable, we have $\max_{D'_j\in\mathcal{D}'} r(\text{Left}(D'_j)) < \min_{D'_i\in\mathcal{D}'} l(\text{Right}(D'_i))$. This implies that $\mathcal{D}'$ is also a subset of $\mathcal{D}^{(k)}$, where $x_k = \max_{D'\in\mathcal{D}'} r(\text{Left}(D'))$. Hence, $\omega(\mathcal{D})$ must be equal to $\omega(\mathcal{D}^{(k)})$ for a specific k. □

By Theorem 1, we can get $\omega(\mathcal{D})$ by computing $\omega(\mathcal{D}^{(k)})$ for $1 \le k \le |X|$. Vialette shows that $\omega(\mathcal{D}^{(k)})$ can be computed by finding a maximum independent set of corresponding trapezoid graphs in $O(n\log n)$ time (Felsner et al., 1997), so his algorithm runs in $O(n^2\log n)$ total time. To improve the complexity, we utilize the dynamic structure of $\mathcal{D}^{(k)}$ by discovering the relationship between $\omega(\mathcal{D}^{(k-1)})$ and $\omega(\mathcal{D}^{(k)})$.

Definition 1. The height of D under $\mathcal{D}^{(k)}$, denoted by $H_k(D)$, is the cardinality of the largest $\{\between\}$-comparable subset of $\mathcal{D}^{(k)}$, whose maximal element must be D under the relation $\between$. That is $H_k(D) = \omega(\{D' \in \mathcal{D}^{(k)} \mid D' \between D\}) + 1$ for $D \in \mathcal{D}^{(k)}$, and $H_k(D) = 0$ for $D \notin \mathcal{D}^{(k)}$.

Obviously, we have $\omega(\mathcal{D}^{(k)}) = \max_{D\in\mathcal{D}^{(k)}} H_k(D)$. If we can compute $H_k(D)$ for each k and D efficiently, then we can get a better upper bound. To achieve this, we first exam the the relationship between $H_k(D)$ and $H_{k-1}(D)$, and show that every non-zero $H_k(D)$ can be computed efficiently.

Lemma 1. *For any k, the $\between$ relation is transitive in $\mathcal{D}^{(k)}$.*

Proof: Given any three 2-intervals $D_1, D_2, D_3 \in \mathcal{D}^{(k)}$, where $D_1 \between D_2$ and $D_2 \between D_3$, we are going to prove that $D_1 \between D_3$. Let $D_1 = (I_1, J_1)$, $D_2 = (I_2, J_2)$ and $D_3 = (I_3, J_3)$, then by the definition of $\between$, we have $I_1 < I_2$ and $I_2 < I_3$, so

$$I_1 < I_3. \tag{1}$$

For the same reason, that is $J_1 < J_2$ and $J_2 < J_3$, we have

$$J_1 < J_3. \tag{2}$$

Because D_3 is in $\mathcal{D}^{(k)}$, we have (by definition of $\mathcal{D}^{(k)}$)

$$r(I_3) \le x_k, \tag{3}$$

and also because D_1 is in $\mathcal{D}^{(k)}$, we have (by definition of $\mathcal{D}^{(k)}$)

$$l(J_1) > x_k. \tag{4}$$

From (3) and (4), we get

$$r(I_3) < l(J_1), \tag{5}$$

from (5) we get

$$I_3 < J_1. \tag{6}$$

Combining (1), (2) and (6), we have

$$I_1 < I_3 < J_1 < J_3. \tag{7}$$

So we know that D_1 and D_3 are $\between$-comparable due to the definition of $\between$, that is

$$D_1 \between D_3. \tag{8}$$

So the $\between$ relation is transitive in the set $\mathcal{D}^{(k)}$ for any fixed k. □

Theorem 2. *For $D \in \mathcal{D}^{(k-1)}$, after the sweep-line goes from x_{k-1} to x_k, the height of D decreases at most by one, or remains the same, i.e. $H_{k-1}(D) - 1 \le H_k(D) \le H_{k-1}(D)$.*

Proof: First, $H_k(D) = h > 1$ if and only if there exists a $D' \in \mathcal{D}^{(k)}$ so that $H_k(D') = h - 1$ and $D' \between D$ (due to Lemma 1). Since no 2-interval in $P^{(k)}$ can cross D, the height of D will not increase. Let $\{D_1, D_2, \ldots, D_{h-1}, D\}$ be a $\{\between\}$-comparable subset of $\mathcal{D}^{(k-1)}$, where $h = H_{k-1}(D)$. Because no pair of 2-intervals in $Q^{(k)}$ are $\between$-comparable, at least $h-1$ elements from the previous set will be preserved in $\mathcal{D}^{(k)}$, hence $H_k(D) \geq h - 1 = H_{k-1}(D) - 1$. □

Let $C_h^{(k)} = \{D \in \mathcal{D}^{(k)} \mid H_k(D) = h\}$. The key issue is to test efficiently for a 2-interval $D \in C_h^{(k-1)}$ whether there exists a $D' \in C_{h-1}^{(k)}$ such that $D' \between D$. In other words, we have to check whether the following set is empty

$$\left\{D' \in C_{h-1}^{(k)} \mid \mathit{r}(\mathrm{Left}(D')) < \mathit{l}(\mathrm{Left}(D)) \text{ and } \mathit{r}(\mathrm{Right}(D')) < \mathit{l}(\mathrm{Right}(D))\right\}.$$

Alternatively, it is to test whether

$$\min\left\{\mathit{r}(\mathrm{Left}(D')) \mid D' \in C_{h-1}^{(k)} \text{ and } \mathit{r}(\mathrm{Right}(D')) < \mathit{l}(\mathrm{Right}(D))\right\} < \mathit{l}(\mathrm{Left}(D)).$$

This property could be tested efficiently by a merge-like process, when the 2-intervals in $C_{h-1}^{(k)}$ are sorted by the key $\mathit{r}(\mathrm{Right}(D'))$, and those in $C_h^{(k-1)}$ are sorted by $\mathit{l}(\mathrm{Right}(D))$. By the discussion above, we have following procedure FALLDOWN(k) to compute $H_k(D)$ from $H_{k-1}(D)$ for $D \in \mathcal{D}^{(k-1)}$ in $O(|\mathcal{D}^{(k-1)}| + |\mathcal{D}^{(k)}|)$ time. For a fixed k that $0 < k \leq |X|$, the procedure partitions $\mathcal{D}^{(k)}$ into $|\mathcal{D}^{(k)}|$ 2-interval set $C_1^{(k)}$, $C_2^{(k)}, \ldots, C_{|\mathcal{D}^{(k)}|}^{(k)}$.

FALLDOWN(k) over all k can be implemented in $O(n \log n + \sum|\mathcal{D}^{(k)}|)$ time.

Procedure 3.1 FALLDOWN(k)

Next, we will give a procedure to compute $H_k(D)$ for $D \in P^{(k)}$. From the fact that there is not any pair of 2-intervals in $P^{(k)}$ that are $\between$-comparable, the height of $D_j \in P^{(k)}$ under $\mathcal{D}^{(k)}$ can be calculated by $H_k(D_j) = \max_{D_i \between D_j} H_k(D_i) + 1$ if there is at least one $D_i \in \mathcal{D}^{(k)} \setminus P^{(k)}$ which cross D_j. Otherwise, $H_k(D_j) = 1$.

Definition 2. For each $0 < h \leq |\mathcal{D}^{(k)}|$, define the boundary with respect to y,

$$B_h^{(k)}(y) = \min_{D \in C_h^{(k)}} \{\mathit{r}(\mathrm{Left}(D)) \mid \mathit{r}(\mathrm{Right}(D)) < y\}. \tag{9}$$

Lemma 2. *For any* $0 < h_1 < h_2 \leq |\mathcal{D}^{(k)}|$, *we have* $B_{h_1}^{(k)}(y) < B_{h_2}^{(k)}(y)$.

Proof: For a fixed k and y, let D_{h_2} denote the 2-interval from $C_{h_2}^{(k)}$ which minimizes $B_{h_2}^{(k)}$ in Eq. (9). Since $h_2 > h_1$, there must be at leat one $D_{h_1} \in C_{h_1}^{(k)}$ that $D_{h_1} \between D_{h_2}$. Since $\mathit{r}(\mathrm{Right}(D_{h_1})) < \mathit{r}(\mathrm{Right}(D_{h_2})) < y$, therefore, we have $B_{h_1}^{(k)}(y) \leq \mathit{r}(\mathrm{Left}(D_{h_1})) < \mathit{r}(\mathrm{Left}(D_{h_2})) = B_{h_2}^{(k)}(y)$. □

Procedure 3.2 JUMPUP(k)

The procedure JUMPUP(k) is used for computing the heights of 2-intervals in $P^{(k)}$ under $\mathcal{D}^{(k)}$. The sorting process in Line 2 and the merge process in Lines 12–14 can be done in $O(|\mathcal{D}^{(k)}|)$ time, if we first sort each $P^{(k)}$ in a global stage. Line 6 can

```
**Function**: Compute $H_k(D)$ for every $D \in \mathcal{D}^{(k-1)}$ from $H_{k-1}(D)$.
**Notation**:Let $C_h^{(k)}[i]$ denote the $i$th element in sorted $C_h^{(k)}$.
Sort the 2-intervals in each $C_h^{(k-1)}$ according to the key $\mathscr{l}(\text{Right}(D))$.
$C_1^{(k)} \leftarrow C_1^{(k-1)} \setminus Q^{(k)}$, sort it by the key $\mathscr{r}(\text{Right}(D))$
for $h \leftarrow 2$ to $|\mathcal{D}^{(k-1)}|$ do
    $C_h^{(k)} \leftarrow C_h^{(k-1)}$;  Decreased $= \emptyset$;  $j \leftarrow 1, i \leftarrow 1$;  LEFTMOST $\leftarrow +\infty$;
    while $j \le |C_h^{(k-1)}|$ do
        while $\mathscr{r}(\text{Right}(C_{h-1}^{(k)}[i])) < \mathscr{l}(\text{Right}(C_h^{(k-1)}[j]))$ do
            LEFTMOST $\leftarrow \min\left\{\text{LEFTMOST}, \mathscr{r}(\text{Left}(C_{h-1}^{(k)}[i]))\right\}$
            $i \leftarrow i + 1$
        end while
        if LEFTMOST $< \mathscr{l}(\text{Left}(C_h^{(k-1)}[j]))$ then
            Do nothing.    // the height of $C_h^{(k-1)}[j]$ remains the same
        else
            Decreased $\leftarrow$ Decreased $\cup \left\{C_h^{(k-1)}[j]\right\}$
            $C_h^{(k)} \leftarrow C_h^{(k)} \setminus C_h^{(k-1)}[j]$
        end if
        $j \leftarrow j + 1$
    end while
    $C_{h-1}^{(k)} \leftarrow C_{h-1}^{(k)} \cup$ Decreased
    Sort the 2-intervals in $C_h^{(k)}$ by the key $\mathscr{r}(\text{Right}(D))$
end for
```

```
Calculate the Heights of $P^{(k)}$ under $\mathcal{D}^{(k)}$
set $B_h^{(k)} \leftarrow \infty$ for each $1 \le h \le |\mathcal{D}^{(k)}|$ and set $B_0^{(k)} \leftarrow -\infty$
sort 2-intervals in $\mathcal{D}^{(k)}$ according to key$(D)$, where

    $$\text{key}(D) = \begin{cases} \mathscr{r}(\text{Right}(D)), & \text{if } D \in \mathcal{D}^{(k)} \setminus P^{(k)} \\ \mathscr{l}(\text{Right}(D)), & \text{if } D \in P^{(k)} \end{cases}$$

If there is a tie, let the elements in $P^{(k)}$ go first.
for $i \leftarrow 1$ to $|\mathcal{D}^{(k)}|$ do
    let $D$ be $i$th element in $\mathcal{D}^{(k)}$
    if $D \in P^{(k)}$ then
        find the largest $h$ such that $B_h^{(k)} < \mathscr{l}(\text{Left}(D))$
        $H_k(D) \leftarrow h + 1$
    else
        let $h \leftarrow H_k(D)$, then set $B_h^{(k)} \leftarrow \min\{B_h^{(k)}, \mathscr{r}(\text{Left}(D))\}$
    end if
end for
for each $0 < h \le |\mathcal{D}^{(k)}|$ do
    $C_h^{(k)} \leftarrow C_h^{(k)} \cup \left\{H_k(D) = h \mid D \in P^{(k)}\right\}$
end for
```

be implemented by binary search (Lemma 2) in $O(\log|\mathcal{D}^{(k)}|)$ time. Thus the total complexity for all JUMPUP(k) is

$$\begin{aligned}\sum O\big(|\mathcal{D}^{(k)}| + |P^{(k)}|\log|\mathcal{D}^{(k)}|\big) &= O\Big(\sum|\mathcal{D}^{(k)}| + \sum|P^{(k)}|\log n\Big)\\ &= O\Big(\sum|\mathcal{D}^{(k)}| + n\log n\Big).\end{aligned}$$

Combining procedures FALLDOWN(k) and JUMPUP(k), we finally obtain Algorithm 3.3. It is easy to see that the space complexity is $O(n)$. By the equation $\sum|\mathcal{D}^{(k)}| = \mathcal{L}$, we have the total time complexity $O\ (n\log n + \mathcal{L})$.

The correctness of Algorithm 3.3 is based on the following facts:

Lemma 3. *No 2-interval in $P^{(k)}$ will cross any element in $\mathcal{D}^{(k)}$.*

Corollary 1. *The height of a 2-interval $D \in \mathcal{D}^{(k)} \setminus P^{(k)}$ under $\mathcal{D}^{(k)}$ is the same to its height under $\mathcal{D}^{(k)}\setminus P^{(k)}$.*

Theorem 3. *Procedure* FALLDOWN*(k) correctly computes the heights of 2-intervals in $\mathcal{D}^{(k)}\setminus P^{(k)}$ under $\mathcal{D}^{(k)}$.*

Algorithm 3.3 $\{\between\}$-STRUCTURED 2-INTERVAL PATTERN

```
sort the 2-intervals and their endpoints
calculate $P^{(k)}$ and $Q^{(k)}$ for each $1 \le k \le |X|$
set $C_h^{(0)} = \emptyset$ for all $1 \le h \le n$
for $k \leftarrow 1$ to $|X|$ do
    call Procedure FALLDOWN($k$)    // compute $H_k(D)$ for $D \in \mathcal{D}^{(k)} \setminus P^{(k)}$
    call Procedure JUMPUP($k$)      // compute $H_k(D)$ for $D \in P^{(k)}$
    $\omega(\mathcal{D}^{(k)})$ equals to the largest $h$ such that $C_h^{(k)} \neq \emptyset$, or zero if $\mathcal{D}^{(k)} = \emptyset$
end for
return max $\omega(\mathcal{D}^{(k)})$
```

Proposition 1. *The* $\{\between\}$-STRUCTURED 2-INTERVAL PATTERN *problem can be solved in $O(n\log n + \mathcal{L})$ time.*

For the case of disjoint support, it can be transformed to the problem of finding a maximum clique in circle graphs (Golumbic, 1980). Each 2-interval maps to a vertex, and two vertices are adjacent if and only if their corresponding 2-interval are $\between$-comparable. By the result of Masuda et al. (1990), we have

Proposition 2. *The* DISJOINT SUPPORT $\{\between\}$-STRUCTURED 2-INTERVAL PATTERN *problem can be solved in $O\ (n\log n + \min\{m, dn\})$ time, where m is the number of 2-interval pairs that are $\between$-comparable.*

4 Improved algorithm for $\{<, \sqsubset\}$-structured pattern

In this section, we will give an $O(n\log n + dn)$ algorithm for $\{<, \sqsubset\}$-STRUCTURED 2-INTERVAL PATTERN problem, and it is $O(n\log n + \mathcal{L})$ for the case that no 2-intervals share the same rightmost endpoint. Our algorithm improves the $O(n^2)$ upper bound given by Vialette (2004), in the sense that if we parameterize d, which is rather small when dealing with most of RNA sequences. What's more, the case in this section is also the exact model of most of RNA sequences. The algorithm is similar to the maximum weighted independent set algorithm for circle graphs which is proposed recently by Valiente (2003).

Definition 3. Let $\mathcal{D}[z_1, z_2]$ represent the 2-intervals lie in $[z_1, z_2]$, where $z_1 < z_2$, i.e. $\mathcal{D}[z_1, z_2] = \{D \in \mathcal{D} \mid z_1 \leq l(\text{Left}(D)) < r(\text{Right}(D)) \leq z_2\}$.

Definition 4. Let $\alpha(\mathcal{D})$ be the cardinality of largest $\{<, \sqsubset\}$-comparable subset of $\mathcal{D}$. Take $\alpha[z_1, z_2]$ instead of $\alpha(\mathcal{D}[z_1, z_2])$ for short. Let $\alpha(D)$ denote the cardinality of largest $\{<, \sqsubset\}$-comparable subset of $\mathcal{D}[l(\text{Left}(D)), r(\text{Right}(D))]$), with the constraint that D must be in that largest cardinality subset.

Lemma 4. $\alpha(D) = 1 + \alpha[r(\text{Left}D) + 1, l(\text{Right}D) - 1]$.

Lemma 5. *Let* $\alpha'[z_1, z_2] = \max\{\alpha\,[z_1, l(\text{Left}D) - 1] + \alpha(D)\}$, *where* $D \in \mathcal{D}[z_1, z_2]$ *and* $r(\text{Right}D)= z_2$. *We have* $\alpha[z_1, z_2] = \max\left\{\alpha[z_1, z_2 - 1], \alpha'[z_1, z_2]\right\}$.

Combining the two Lemmas above, we have a dynamic programming algorithm working in $O(n^2)$ time. To achieve a tighter bound, we can first compute $\alpha(D)$ for every $D \in \mathcal{D}$ in the order of their lengths (see next paragraph), and then calculate $\alpha(\mathcal{D}) = \alpha[x_1, x_{|X|}]$ in $O(|X|)$ time by Lemma 5 .

To compute the value of $\alpha(D_i)$, we only have to calculate $\alpha[r(\text{Left}(D_i)) + 1, l(\text{Right}(D_i)) - 1]$ due to Lemma 4. For a specific D_i, let $\beta[z]$ denote $\alpha[r(\text{Left}(D_i)) + 1, z]$, then after slightly modifying Lemma 5, we have

$$\beta[z] = \max_{D'}\{\beta[z-1], \beta[l(\text{Left}(D')) - 1] + \alpha(D')\}$$

where D' goes through all 2-intervals which satisfies $r(\text{Right}(D')) = z$ and $l(\text{Left}(D')) \geq r(\text{Left}(D_i)) + 1$. The special case is that

$$\beta[z] = 0 \quad \text{for } z \leq r(\text{Left}(D_i)) + 1.$$

Since the goal is to find out the value of $\alpha(D_i) = \beta[l(\text{Right}(D_i)) - 1] + 1$, we only have to compute $\beta[z]$ for $r(\text{Left}(D_i)) < z < l(\text{Right}(D_i))$. So only $O(\text{Length}(D_i))$ temporary space is required. The computational cost for an entry $\beta[z]$ is in proportion to the number of 2-intervals whose rightmost endpoint is z and its length is smaller than Length(D_i). All these 2-intervals can be enumerated efficiently by constructing a sorted list List(x_k) for each $x_k \in X$ in total $O(n\log n)$ time at the preprocessing stage, where List(x_k) contains all 2-intervals whose rightmost endpoint is x_k, and the 2-intervals in each list is sorted by their lengths.

Theorem 4. *For an 2-interval $D_i \in \mathcal{D}$, if we know the values of $\alpha(D')$ for all $D' \sqsubset D_i$ that* LengthD' < LengthD_i, *then we can compute $\alpha(D_i)$ in $O(\text{Length}D_i + m_i)$ time and only use $O(\text{Length}D_i)$ temporary space, where m_i is the number of 2-intervals nested in D_i.*

The total complexity is $O(n\log n + \mathcal{L} + \sum m_i + |X|)$. It is not too difficult to see that $\sum m_i \le dn$. If no 2-intervals share the same rightmost endpoint, then it is easy to see that $m_i \le \text{Length}(D_i)$, hence we have $\sum m_i \le \mathcal{L}$ in this case.

Proposition 3. *The* $\{<, \sqsubset\}$-Structured 2-Interval Pattern *problem can be solved in $O(n\log n + dn)$ time and $O(n)$ space. If no 2-intervals share the same rightmost endpoint, then the time complexity can be improved to $O(n\log n + \mathcal{L})$.*

5 Improved algorithm for $\{\sqsubset, \between\}$-structured pattern

In this section, we will give an $O(n\log n + \mathcal{L})$ algorithm for the model $R = \{\sqsubset, \between\}$ with disjoint support. This improves the $O(n^2\sqrt{n})$ algorithm given by Blin et al. (2004).

In the case with disjoint support, define $\mathtt{l}(D)$ and $\mathtt{r}(D)$ to be the left and right endpoints respectively for a 2-interval $D \in \mathcal{D}$, i.e. $\mathtt{l}(D) = \mathtt{l}(\text{Left}(D)) = \mathtt{r}(\text{Left}(D))$ and $\mathtt{r}(D) = \mathtt{l}(\text{Right}(D)) = \mathtt{r}(\text{Right}(D))$.

Lemma 6. *Let $\mathcal{D}'$ be a $\{\sqsubset, \between\}$-comparable subset of $\mathcal{D}$, then*

$$\max_{D'_j \in \mathcal{D}'} \mathtt{l}(D'_j) < \min_{D'_i \in \mathcal{D}'} \mathtt{r}(D'_i).$$

Proof: Otherwise, there must be two 2-intervals D'_j and D'_i such that $\mathtt{l}(D'_j) \ge \mathtt{r}(D'_i)$. If $\mathtt{l}(D'_j) = \mathtt{r}(D'_i)$, then D'_j and D'_i are not comparable; if $\mathtt{l}(D'_j) > \mathtt{r}(D'_i)$, then we have $D'_i < D'_j$. Both cases lead to contradictions, because D'_j and D'_i should be either $\sqsubset$-comparable or $\between$-comparable! □

Define $\varphi(\mathcal{D})$ to be the largest largest $\{\sqsubset, \between\}$-comparable subset of $\mathcal{D}$ with disjoint support. Similar to Section 3, we have $\varphi(\mathcal{D}) = \max \varphi(\mathcal{D}^{(k)})$. Blin et al. calculate each $\varphi(\mathcal{D}^{(k)})$ by finding the maximum cardinality matching in the corresponding bipartite graphs. We still apply this idea, but the complexity is improved by discovering the dynamic structure of the bipartite graphs.

Definition 5. For a fixed $0 < k \le |X|$, let $G^{(k)} = (U^{(k)}, V^{(k)}, E^{(k)})$ be a bipartite graph corresponding to $\mathcal{D}^{(k)}$ defined as follows: $U^{(k)} = \{x \in X \mid x \le x_k\}$, and $V^{(k)} = \{x \in X \mid x > x_k\}$, the edges $E^{(k)} = \{\langle \mathtt{l}(D), \mathtt{r}(D)\rangle \mid D \in \mathcal{D}^{(k)}\}$.

Obviously, a maximum matching of $G^{(k)}$ corresponds to a maximum cardinality $\{\sqsubset, \between\}$-comparable subset of $\mathcal{D}^{(k)}$. Let $M^{(k)}$ be a maximum matching in $G^{(k)}$.

Theorem 5. *The cardinality of maximum matching in $G^{(k)}$ and $G^{(k-1)}$ differs at most one, i.e.* $|M^{(k-1)}| - 1 \le |M^{(k)}| \le |M^{(k-1)}| + 1$.

Proof: Define the differences of $E^{(k)}$ and $E^{(k-1)}$ by

$$F_+^{(k)} = E^{(k)} \setminus E^{(k-1)} = \{\langle x_k, x_j\rangle \mid x_j = \mathscr{r}(D) \text{ for } D \in \mathcal{D} \text{ where } \mathscr{l}(D) = x_k\},$$

$$F_-^{(k)} = E^{(k-1)} \setminus E^{(k)} = \{\langle x_i, x_k\rangle \mid x_i = \mathscr{l}(D) \text{ for } D \in \mathcal{D} \text{ where } \mathscr{r}(D) = x_k\}.$$

Let $M = M^{(k-1)} \setminus \{\langle x_i, x_k\rangle\}$ if there is an edge $\langle x_i, x_k\rangle \in F_-^{(k)}$, otherwise $M = M^{(k-1)}$. Obviously, M is a matching of $G^{(k)}$, so $|M^{(k-1)}| - 1 \le |M| \le |M^{(k)}|$. The second part of inequalities can be proved in the same way. Let $M' = M^{(k)} \setminus \{\langle x_k, x_j\rangle\}$ if there is an edge $\langle x_k, x_j\rangle \in F_+^{(k)}$, otherwise $M' = M^{(k)}$. Now M' is a matching of $G^{(k-1)}$, so $|M^{(k)}| - 1 \le |M'| \le |M^{(k-1)}|$. □

Theorem 6 (Hopcroft and Karp (1973)). *Let M_1 and M_2 be two matchings, if $|M_1| = s$, $|M_2| = r$ and $r > s$, then $M_1 \oplus M_2$ contains at least $r - s$ vertex disjoint augmenting paths relative to M_1. Where the operation $\oplus$ means*

$$M_1 \oplus M_2 = \{e \mid e \in M_1 \text{ and } e \notin M_2\} \cup \{e \mid e \notin M_1 \text{ and } e \in M_2\}.$$

By Theorems 5 and 6, we have Algorithm 5.1 to find the maximum matching of $G^{(k)}$ for each $0 < k \le |X|$ efficiently. Since the number of edges is at most $|\mathcal{D}|$, so the space complexity is $O(n)$. It is easy to see that the time complexity from Line 4 to Line 9 is $O(\sum|E^{(k)}|) = O(\sum|\mathcal{D}^{(k)}|) = O(\mathcal{L})$. Thus, our algorithm is worst-case quadratic, which improves the previous best known upper bound $O(n^2\sqrt{n})$.

Algorithm 5.1 DISJOINT SUPPORT $\{\sqsubset, \between\}$-STRUCTURED 2-INTERVAL PATTERN

```
1: sort the endpoints of $\mathcal{D}$
2: calculate $F_+^{(k)}$ and $F_-^{(k)}$ for every $0 < k \le |X|$
3: $E \leftarrow \emptyset$, $M \leftarrow \emptyset$
4: for $k \leftarrow 1$ to $|X|$ do
5:     $E \leftarrow E \setminus F_-^{(k)}$, $M \leftarrow M \setminus F_-^{(k)}$
6:     if there exist an augmenting path $P \in E$ relative to $M$, then $M \leftarrow M \oplus P$
7:     $E \leftarrow E \cup F_+^{(k)}$
8:     if there exist an augmenting path $P \in E$ relative to $M$, then $M \leftarrow M \oplus P$
9: end for
```

Proposition 4. *The* DISJOINT SUPPORT $\{\sqsubset, \between\}$-STRUCTURED 2-INTERVAL PATTERN *problem can be solved in $O(n \log n + \mathcal{L})$ time.*

6 Conclusions

In this paper, we give several improved algorithms for different models of 2-INTERVAL PATTERN problem. The case of $R = \{\between\}$ and $R = \{\sqsubset, \between\}$ with disjoint support are solved both in $O(n\log n + \mathcal{L})$ time by sweep-line based method. An $O(n\log n + dn)$ algorithm is given to solve the case $R = \{<, \sqsubset\}$, which is similar to previous known maximum independent set algorithm for circle graphs. All of our algorithms require only linear space.

Acknowledgments Thanks to Hong Zhu, Binhai Zhu, Yunfeng Tao and the reviewers for their helpful comments. Thanks to Prof. Yong Yu, the advisor of the undergraduate program in Department of Computer Science and Engineering, Shanghai Jiao Tong University, for his support on this work.

References

Alber J, Gramm J, Guo J, Niedermeier R (2004) Computing the similarity of two sequences with nested arc annotations. Theor Comput Sci 312(2–3):337–358

Bar-Yehuda R, Halldórsson MM, Naor J, Shachnai H, Shapira I (2002) Scheduling split intervals. In: Proceedings of the 13th annual ACM-SIAM symposium on discrete algorithms, pp 732–741

Blin G, Fertin G, Vialette S (2004) New results for the 2-interval pattern problem. In: Combinatorial pattern matching, 15th annual symposium, CPM 2004, proceedings, Springer, pp 311–322. ISBN 3-540-22341-X

Crochemore M, Hermelin D, Landau GM, Vialette S (2005) Approximating the 2-interval pattern problem. In: ESA, pp 426–437

Evans PA (1999) Finding common subsequences with arcs and pseudoknots. In: Crochemore M, Paterson M (eds) Combinatorial pattern matching, 10th annual symposium, CPM 99, proceedings, Springer, pp 270–280. ISBN 3-540-66278-2

Felsner S, Müller R, Wernisch L (1997) Trapezoid graphs and generalizations, geometry and algorithms. Discr Appl Math 74(1):13–32

Golumbic M (1980) Algorithmic graph theory and perfect graphs. Academic Press, New York, NY

Gramm J (2004) A polynomial-time algorithm for the matching of crossing contact-map patterns. In: Algorithms in bioinformatics, 4th international workshop, WABI 2004, proceedings, Springer, pp 38–49. ISBN 3-540-23018-1

Hopcroft JE, Karp RM (1973) An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J Comput (4)

Jiang T, Lin G, Ma B, Zhang K (2004) The longest common subsequence problem for arc-annotated sequences. J. Discr Algor 2(2):257–270

Masuda S, Nakajima K, Kashiwabara T, Fujisawa T (1990) Efficient algorithms for finding maximum cliques of an overlap graph. Networks 20:157–171

Micali S, Vazirani V (1980) An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In: Proceedings of the 21st annual symposium on foundation of computer science, IEEE, pp 17–27

Valiente G (2003) A new simple algorithm for the maximum-weight independent set problem on circle graphs. In: Algorithms and computation, 14th international symposium, ISAAC 2003, proceedings, Springer, pp 129–137. ISBN 3-540-20695-7

Vialette S (2004) On the computational complexity of 2-interval pattern matching problems. Theor Comput Sci 312(2–3):223–249