Approximating The Anticover of a String

Amihood Amir, Itai Boneh and Eitan Kondratovsky

Bar Ilan University

Anticover – Definition (Alzamel et al. CPM 2020)

- Given a string $S[1 \dots n]$, and an integer parameter k, a k-anticover of S is a set of distinct substrings of size k that together cover every index in S.
- Example:

$S = abacbac$ $S[1 \dots 3], S[4 \dots 6]$ is not a 3-anticover since $S[7]$ is not covered.

$S = abacbac$ $S[1 \dots 3], S[2 \dots 4], S[5 \dots 7]$ is not a 3-anticover since $\boldsymbol{S[2 \dots 4] = S[5 \dots 7]} = bac$

$S = abacbac$ $S[1 \dots 3], S[3 \dots 5], S[5 \dots 7]$ is a 3-anticover

Anticover - Motivation

- Indicates a lack of structure in a string.
- The ‘opposite’ of covers\period, more likely to be found.
- Possible algorithmic application.

Anticover – Recent Results

Alzamel et al [CPM 2020]:

For $k = 2$:

- a 2-anticover of a string $S[1 \ldots n]$ over an alphabet Σ can be reported in $O(n|\Sigma|)$, if a 2-anticover exists.

For $k \geq 3$,

- finding whether a string $S[1 \ldots n]$ has a k-anticover or not is NP complete.
- $O^*(\min(3^{\frac{n-k}{3}}, \left(\frac{k(k+1)}{2}\right)^{\frac{n}{k+1}}))$ algorithm. Better than naïve.

Anticover – Optimization

- **MaxkAnticover-** Input: S and k. find a set of distinct k length substrings that maximizes the number of covered indexes.

- Example:
- $S = aaaaaab$ does not have a 3-anticover. But we can cover 6 indexes using distinct substings.

$$S = \underline{aaa}\underline{a}\underline{aab}$$

Anticover – Optimization

- **MinRepkAnticover:** Input: S and k. Find a set of k-length substrings that covers every index in S, and minimizes the number of occurrences of the same string
- Example:
  $S = aaabaaaaaa$ can be covered with $aaabaaaaaa$, which uses the string ‘aaa’ 3 times. A **MinRepkAnticover** optimal cover would be : $aaabaaaaaa$, which uses ‘aaa’ twice.

Anticover – Optimization

- **MinAnticover:** Input: a string S, output the minimal integer k such that S has a k-anticover.
- Example:
  $S = abacbac$ has a 3-anticover. It also has a 2-anticover:

$$abacbac$$

S does not have a 1-anticover, so **MinAnticover(S)=2**.

Anticover - Approximation

Optimization variant	Approximation ratio	Time (fixed sized Σ)	Time(Inf integer Σ)
MaxkAnticover	$\frac{1}{2}$	$O(n)$	$O(n \log n)$
MinRepkAnticover	$\log_2 n + 1$	$O(nm \log n)^*$	$O(nm \log n)^*$
MinAnticover	4	$Polynomial$	$Polynomial$

* m denotes the optimal value of MinRepkAnticover.

Maximizing Covered Indexes - Algorithm

We use a **greedy** approach to approximate **MaxkAnticover**.

- Create a **pool** of substrings with $|\mathrm{w}| = k$. Every substring w is covering an interval $[i \ldots i + k - 1]$ of indexes in S.

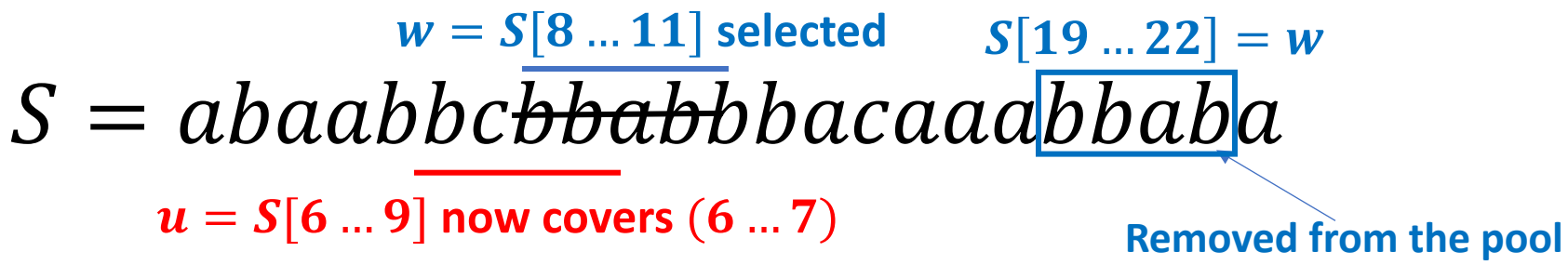
- Select a word from the pool and add it to the cover. Stop when the pool is empty.

$$S = abaabbcbbabbbacaaabbaba$$

w covers $(8, 11)$

Maximizing Covered Indexes – Algorithm

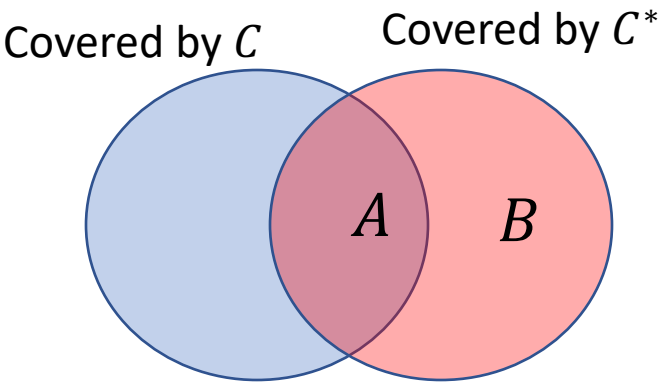
- Greedy selection: pick a word $\boldsymbol{w}$ that covers the maximal number of uncovered indexes and add it to the cover.
- Update the intervals for words **overlapping** with $\boldsymbol{w}$.
- Remove occurrences of $\boldsymbol{w}$ from the pool



Maximizing Covered Indexes - Approximation

Claim: the greedy approach yields a $\frac{1}{2}$ approximation for **MaxkAnticover**.

- C^*- An optimal k-anticover of S.
- C - The output of the greedy algorithm.



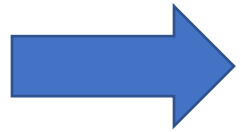
We partition the indexes coved by C^* to two groups:

- A – indexes that are covered by C.
- B – indexes that are not covered by C

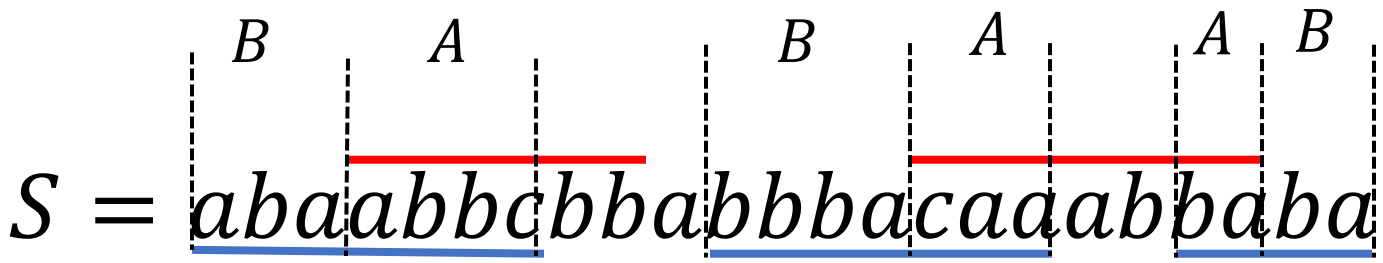
Maximizing Covered Indexes - Approximation

Obviously, $|A| \leq Cov(C)$.
Let w be a word in C^* that covers $x > 0$ indexes in B.

Our algorithm terminated without adding w to the cover

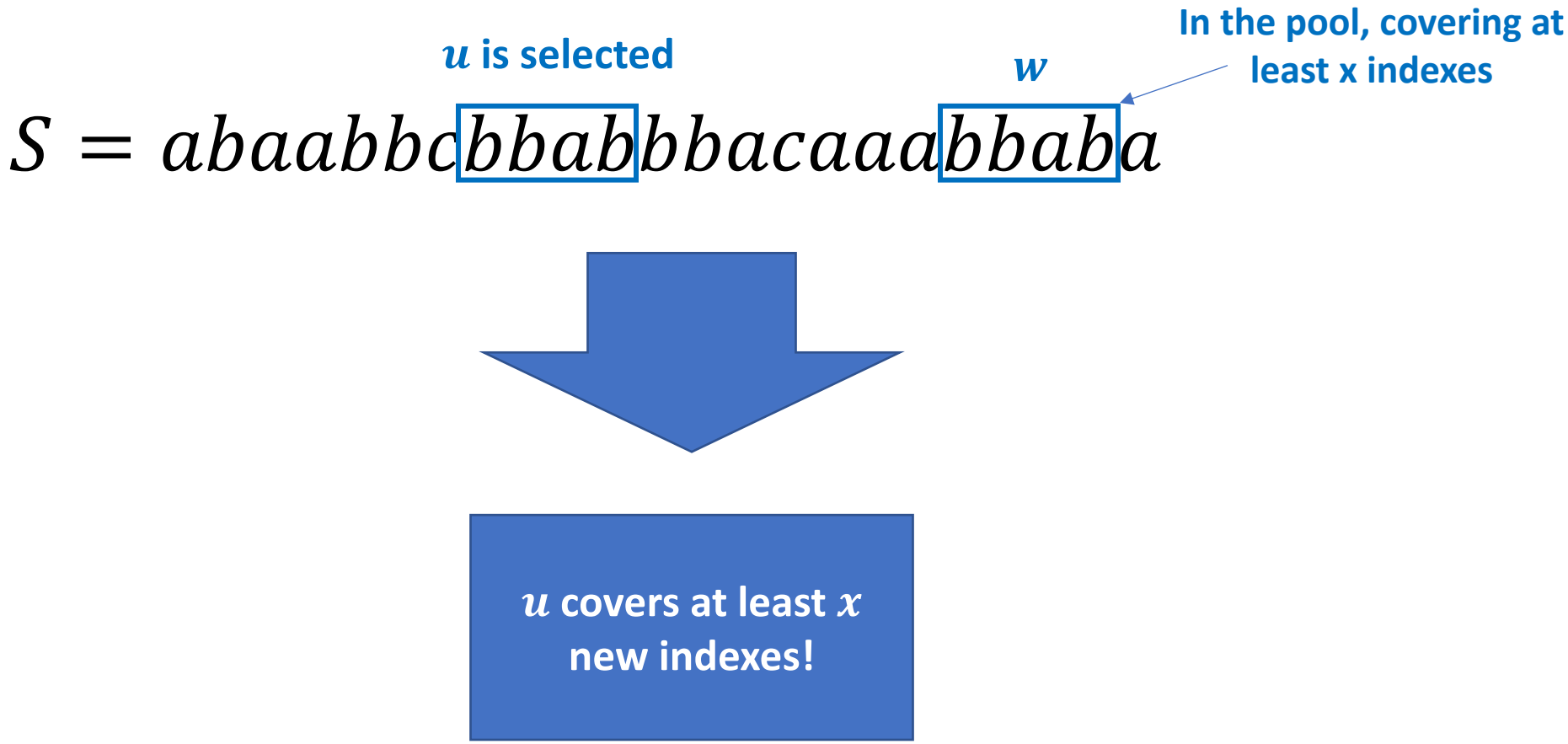


Our cover contains an occurrence u of w.



Maximizing Covered Indexes - Approximation

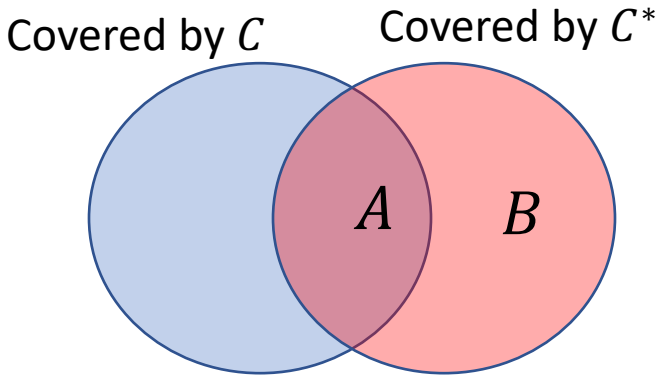
The moment of adding u to the cover:



Maximizing Covered Indexes - Approximation

In conclusion – every word in C^* covering x indexes in B is matched to a word in C covering at least x indexes.

Therefore: $|B| \leq Cov(C)$.



$$Cov(C^*) = |A| + |B| \leq Cov(C) + Cov(C) = 2Cov(C)$$

Maximizing Covered Indexes – Hard Example

- For the following string, our algorithm does not do better than a $\frac{1}{2}$ approximation for **maxkAnticover**:

$$S = aaaaaa \dots\dots\dots aaaaaabbbbbbb \dots\dots\dots\dots\dots\dots bbbbbb$$

- Our algorithm may cover $2k$ indexes:

$$S = aaaaaa \dots\dots\dots aaaaaabbbbbbb \dots\dots\dots\dots\dots\dots bbbbbb$$

- The optimal cover covers $4k - 2$:

$$S = aaaaaa \dots\dots\dots aaaaaabbbbbbb \dots\dots\dots\dots\dots\dots bbbbbb$$

Maximizing Covered Indexes – Experiment

- We ran our approximation on various strings, looking for worst case examples in a brute force manner.
- The $\frac{2k}{4k-2}$ bound comes up, but nothing worst.
- Applying intuitive heuristics for breaking ties seems to achieve a better ratio.

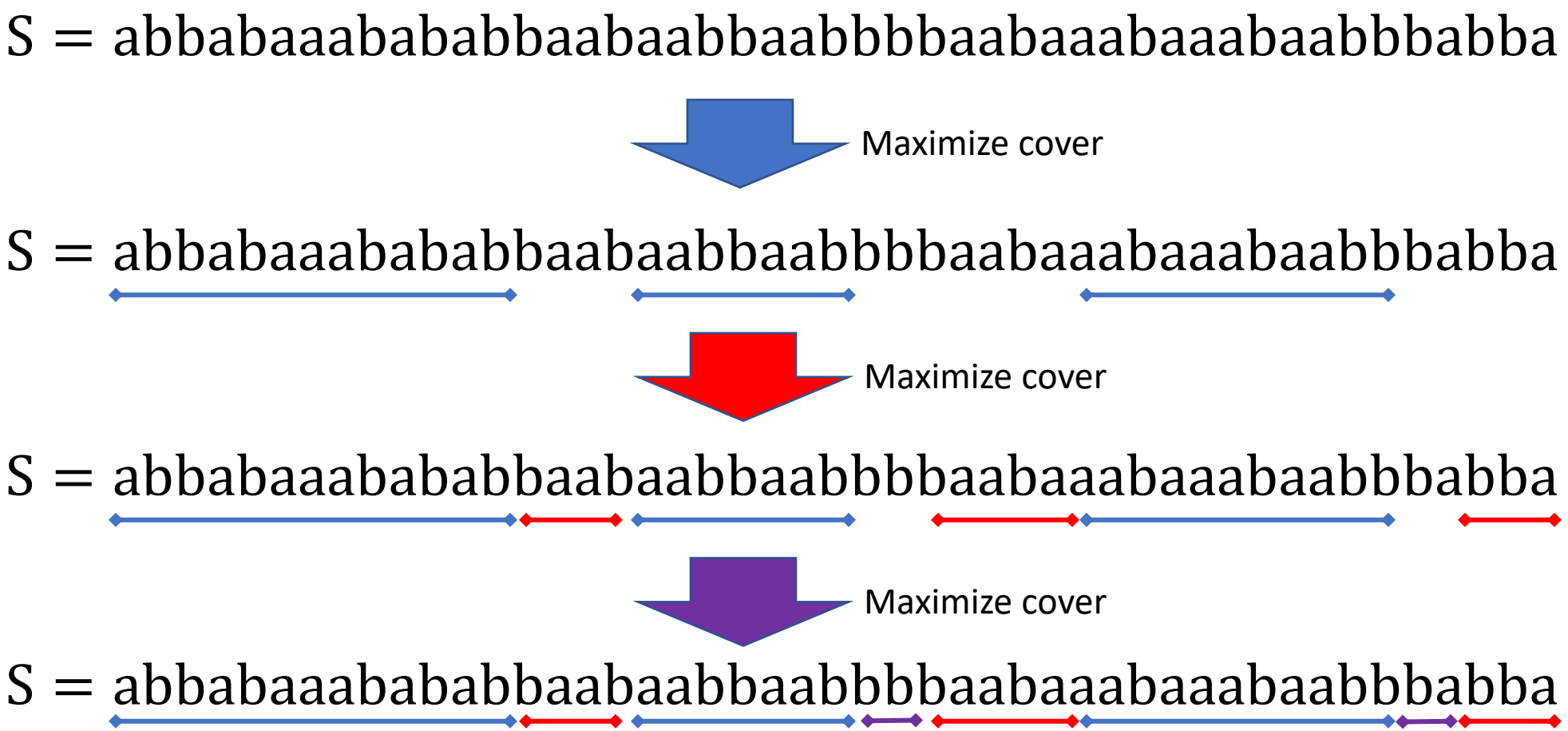
Minimizing repetitions- Algorithm

Observation: Our $\frac{1}{2}$-approxaimtaion for **MaxkAnticover** can be applied to **subsets** of indexes.

$S = \underline{abbabaaa}\underline{babab}\underline{bbaaba}\underline{abbaab}\underline{bbbbaaba}\underline{aaabaaaba}\underline{abbbbabba}$

Maximize the covered **blue** indexes

Minimizing repetitions- Algorithm



Minimizing repetitions- Approximation

- Opt – an optimal cover of S with at most x^* repetitions.
- Opt can be partition into x^* sets $\mathrm{C}_1, \mathrm{C}_2 \ldots C_{x^*}$ of distinct strings.
- Observation: For every subset I of indexes of S there is a set $\mathrm{C_i}$ of distinct strings covering at least $\frac{|I|}{x^*}$ indexes.

Minimizing repetitions- Approximation

- In every iteration, we apply a $\frac{1}{2}$ approximation, so we cover a portion of at least $\frac{1}{2x^*}$ of the remaining indexes.
- In step i of the iteration, there are $n\left(1-\frac{1}{2x^*}\right)^i$ remaining indexes to cover.
- Setting $i = 2x^*\ln(n)$ sets this expression to 1. The next application will surely cover the single remaining indexes.

Minimizing repetitions- Approximation

- Every application of our greedy algorithm adds at most one repetition.
- The number of repetitions is bounded by $2x^* \ln n + 1$, a logarithmic approximation.
- A more careful analysis shows that the number of repetitions is bounded by $x^* \log_2 n + 1$.

Minimizing $\boldsymbol{k}$

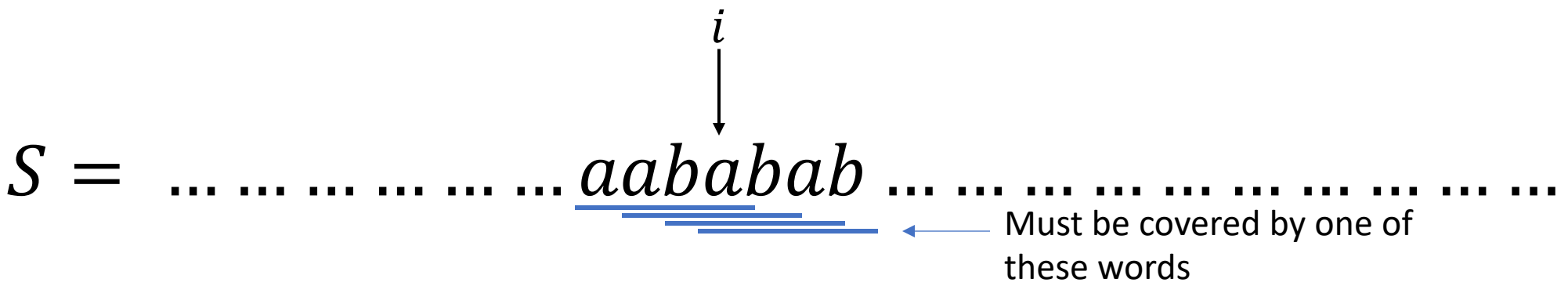
- Definition: The covering set of a word w with length $2k - 1$ is the set of **distinct** subwords of w with length k.

- Example:
  $w = abcdefg$. Covering set: $Cs(w) = \{abcd, bcde, cdef, defg\}$.
- Example:
  $u = aababab$. Covering set: $Cs(w) = \{aaba, abab, baba\}$.

- Note that $|w| = |u| = 4$, but $|Cs(w)| \neq |Cs(u)|$.

Minimizing $\boldsymbol{k}$

- $cw_i = S[i - k + 1 \ldots i + k\ - 1]$
- Observation: In a k −anticover of S, the index i must be covered by a word $w \in Cs(cw_i)$.



Minimizing $\boldsymbol{k}$

- Observation: If there is a word $|w| = 2k - 1$ in S with more than $|Cs(w)|$ occurrences that are k indexes apart from each other – S does not have a k-anticover.

$$w = aababab, |Cs(w)| = 3$$

$$S = \ldots\ldots aababab \ldots\ldots\ldots\ldots aababab \ldots\ldots\ldots\ldots aababab \ldots\ldots aababab$$

Every occurrence must be covered by a **distinct** word in $Cs(w)$**.**

Minimizing $\boldsymbol{k}$ - Algorithm

- k^*- A minimal value such that there is **not** a word w in S with more than $|Cs(w)|$ occurrences that start at least k indexes apart from each other.

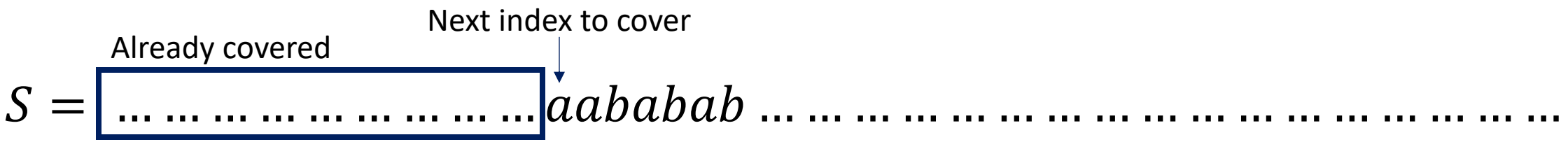
- S does not have a k-anticover for $k < k^*$.

- k^* can be found in polynomial time. Our algorithm uses $k^*$to find a $(4k^* - 1)$-anticover .

Minimizing $\boldsymbol{k}$ - Algorithm

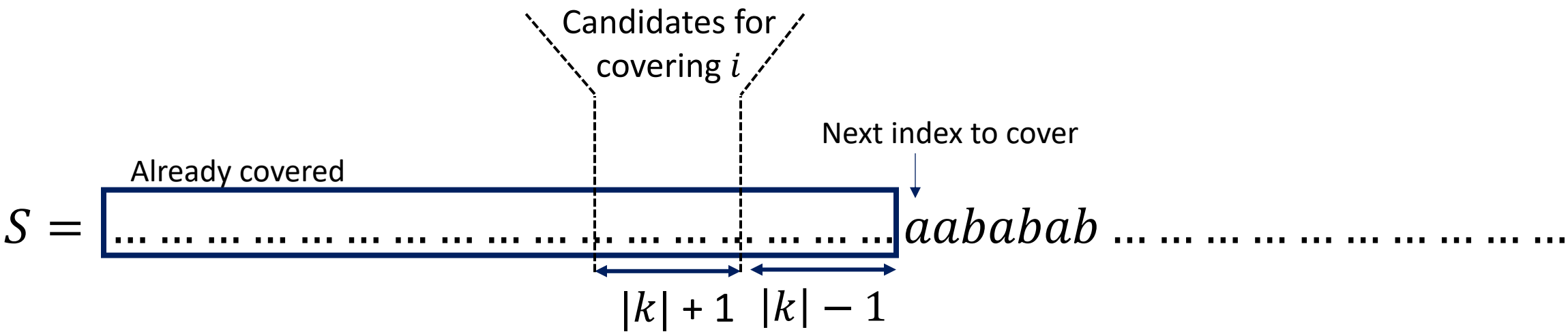
- Start by adding the **first** and the **last** words in S to the $(4k-1)$-anticover.

- We construct the $(4k-1)$- anticover from left to right.



Minimizing $\boldsymbol{k}$ - Algorithm

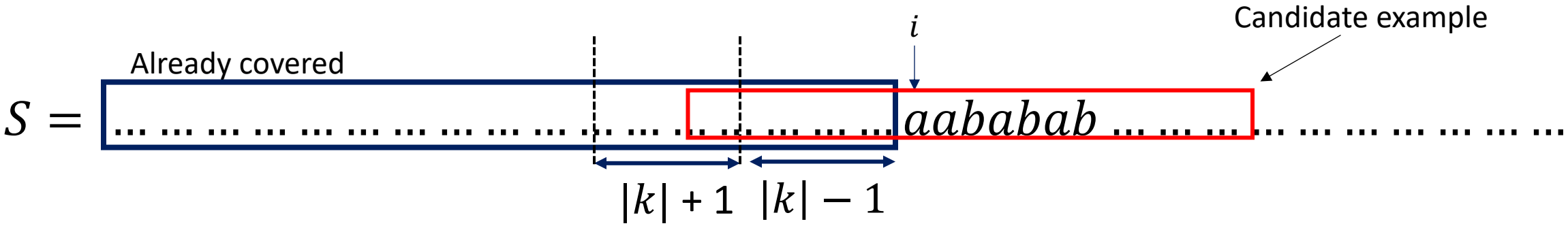
- We arbitrarily choose a word to cover index i among the **available** words starting in $[i\ -(2k-1)\ldots i-(k-1)]$.



Minimizing $\boldsymbol{k}$ – Approximation

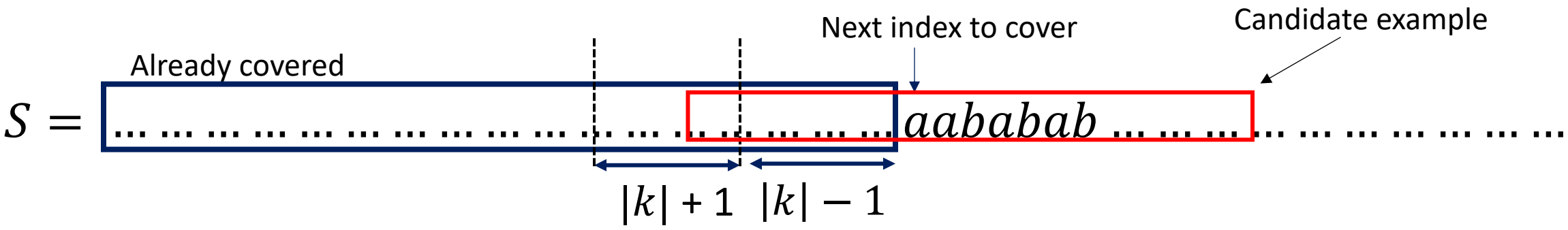
- The only way for our algorithm to fail is if every candidate for covering i has another occurrence that has already been selected.

- Observation: Every candidate contains an occurrence of $\mathrm{cw_i}$



Minimizing $\boldsymbol{k}$ - Approximation

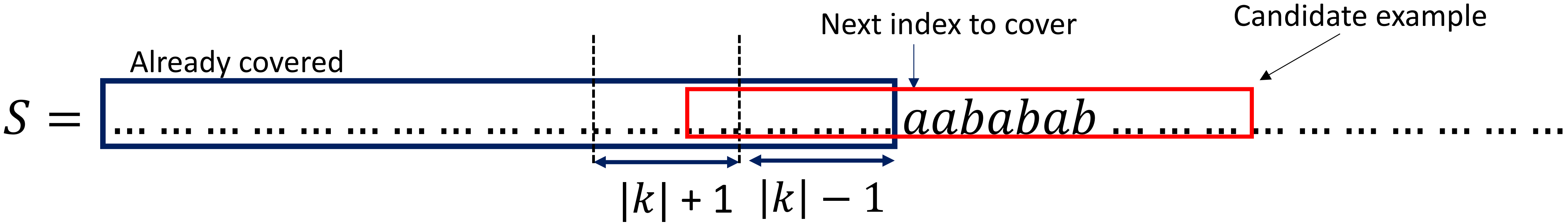
- Observation: There are at least $|\mathrm{Cs}(\mathrm{cw_i})|$ different candidates.
- Observation: The occurrences of $\mathrm{cw_i}$ within the previously selected candidates are at least k indexes apart from each other.



Minimizing $\boldsymbol{k}$ - Approximation

- If **all** the candidates for covering i has already been selected, there are at least $|\mathrm{Cs}(\mathrm{cw_i})|$ occurrences of $\mathrm{cw_i}$ starting within a distance of at least k from each other.

- A contradiction to our selection of k – the algorithm terminate successfully.



Lower Approximation Bounds

Optimization variant	Approximation upper bound	Approximation lower bound
MaxkAnticover	2	$?$
MinRepkAnticover	$\log_2 n + 1$	2
MinAnticover	4	$\frac{4}{3}$

Open problems

Closing gaps:

- For every approximation variant, there is a gap to be closed between the upper and the lower approximation bounds.
- Specifically, we did not rule out the existence of a PTAS for **MaxkAnticover**.

Exploiting Anticoverability-

- Find algorithmic applications for highly anticoverable strings.

Thank You!