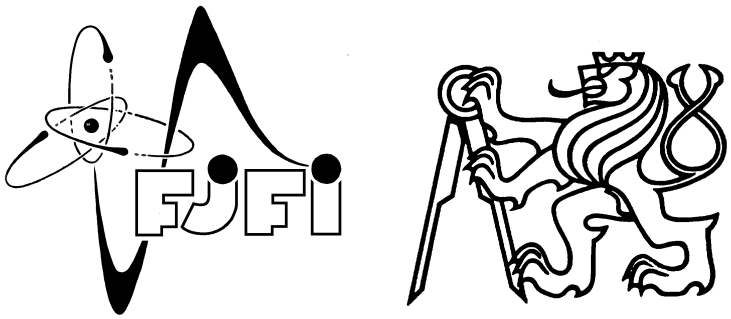
České vysoké učení technické v Praze

Fakulta jaderná a fyzikálně inženýrská

katedra matematiky

obor: Inženýrská informatika

zaměření: Softwarové inženýrství a matematická informatika



Přibližné vyhledávání pomocí bezeztrátového filtrování

Lossless seeds for approximate string matching

Diplomová práce

autor práce: **Bc. Karel Břinda**
vedoucí práce: **Ing. Karel Klouda, Ph.D.**
konzultant: **prof. Gregory Kucherov, Ph.D.**
akademický rok: **2012/13**

Čestné prohlášení

Prohlašuji na tomto místě, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškerou použitou literaturu.

V Praze dne 5. května 2013 .....

Karel Břinda

Acknowledgments

First I would like to express my gratitude and appreciation to my supervisor, Karel Klouda, for regular meetings, patience, valuable ideas, and overall help not only with this diploma thesis.

Then it is a pleasure for me to thank Gregory Kucherov and Guillaume Blin – for the choice of the topic, giving me all materials for study, organization of my stays in Marne-la-Vallée and also for remarkable pieces of advice and ideas.

Finally I would like to thank also to all members of the Theoretical Informatics Group (TIGR) at the FNSPE Czech Technical University in Prague for a pleasant environment and perfect conditions for research.

This work was supported by:

i) the Czech Science Foundation, grant GAČR 13-03538S;

ii) the Grant Agency of the Czech Technical University in Prague, grant No. SGS11/162/OHK4/3T/14;

iii) the Foundation "Nadání Josefa, Marie a Zdeňka Hlávkových".

Název práce:	**Přibližné vyhledávání pomocí bezeztrátového filtrování**
Autor:	Bc. Karel Břinda
Obor:	Inženýrská informatika
Zaměření:	Softwarové inženýrství a matematická informatika
Druh práce:	diplomová práce
Vedoucí práce:	Ing. Karel Klouda, Ph.D. katedra aplikované matematiky Fakulta informačních technologií České vysoké učení technické v Praze
Konzultant:	prof. Gregory Kucherov, Ph.D. Laboratoire d'Informatique Gaspard-Monge Université Paris-Est Marne-la-Vallée

Abstrakt:

Tato práce se zabývá bezeztrátovými seedy, které byly původně navrženy pro účely filtrační fáze při podobnostním vyhledání v DNA. Nejdříve je studována konstrukce seedů pro případ jedné chyby a je uspokojivě vyřešena pomocí hladového algoritmu. Získané výsledky jsou zobecněny pro případ dvou chyb, nicméně je ukázáno, že stejný algoritmus v tomto případě neposkytuje asymptoticky optimální seedy. Dále je představena myšlenka konstrukce seedů pomocí cyklických pravítek. Jsou zmíněna možná zobecnění pro případ více chyb. Na závěr je popsán software vytvořený pro účely této práce, který se nachází na přiloženém CD.

Klíčová slova: přibližné vyhledávání, bezeztrátové seedy, bioinformatika, cyklická pravítka

Title:	**Lossless seeds for approximate string matching**
Author:	Bc. Karel Břinda

Abstract:

The thesis deals with lossless seeds, which were originally proposed for the purposes of the filtration phase in the DNA similarity search. The case of designing seeds for one error is studied first and sufficiently solved using the greedy algorithm. Obtained results are generalized for the case of two errors, nevertheless, it is shown that the same algorithm does not provide asymptotically optimal seeds in this case. Further on, an idea of seed design based on the so-called cyclic rulers is introduced. Possibilities of generalization for the case of more errors are mentioned. At the end the software created for the purposes of this thesis, which is given on the enclosed CD, is described.

Keywords: approximate string matching, lossless seeds, bioinformatics, cyclic rulers

Contents

List of tables

List of figures

List of symbols

Symbol	Description
$\mathbb{N}$	set of natural numbers, $\mathbb{N} = \{0, 1, 2, ...\}$
$\mathbb{Z}$	set of integer numbers, $\mathbb{Z} = \{..., -2, -1, 0, 1, 2, ...\}$
$\#M$	cardinality of the set M
$\binom{n}{k}$	binomial coefficient, $\binom{n}{k} = \frac{n!}{k!(n-k)!}$
$\vec{x}$	column vector, $\vec{x} = \begin{pmatrix} x_0 \\ \vdots \\ x_{n-1} \end{pmatrix}$
$\mathcal{A}$	alphabet
$\mathcal{A}^n$	set of all strings over $\mathcal{A}$ of length n
$\mathcal{A}^*$	set of all finite strings over $\mathcal{A}$
$\mathcal{A}^{\mathbb{N}}$	set of all infinite strings over $\mathcal{A}$
ε	empty string in $\mathcal{A}^*$
S, T	finite strings over $\mathcal{A}$, $S = S_0 S_1 \cdots S_{n-1}$
S^ω	repetition of S infinitely many times, $S^\omega = SSS\cdots$
$\lvert S\rvert$	length of the string S
$\lvert S\rvert_a$	number of occurrences of the letter a in the string S
Q	seed
s	span of a seed
$s(Q)$	span of the seed Q, $s(Q) = \lvert Q\rvert$
w	weight of a seed
$w(Q)$	weight of the seed Q, $w(Q) = \lvert Q\rvert_{\#}$
ℓ	$m - s$ for some (m, k)-problem
M_Q	incidence matrix of the seed Q in an $(m, 2)$-problem
$\mathbb{M}_Q$	k-dimensional incidence matrix of the seed Q in an (m, k)-problem

Introduction

The following text is divided into six chapters.

1. **Preliminaries**
   After a short overview about the topic of this thesis, all necessary notation in the area of combinatorics on words is presented (in particular, free monoids) and spaced seeds are defined. Then a motivation to use them is demonstrated on an example of their application for DNA similarity search.

2. **Lossless seeds survey**
   This chapter recalls some already known results concerning lossless seeds for the case of single seed and one hit.

3. **One error**
   A necessary and sufficient criterion for solvability in the case of one error is presented and it is shown how to find optimal seeds using the greedy algorithm.

4. **Two errors**
   Incidence matrices for seeds in $(m, 2)$-problems are defined and their application for checking the solvability of $(m, 2)$-problems by given seeds is described. Combination of greedy algorithm with this approach provides very good seeds, for low m's even optimal, however, not asymptotically optimal. An approach based on cyclic rulers is introduced including a conjecture that optimal seeds can be obtained using this approach.

5. **More errors**
   Possibilities of generalization of the results from the previous chapter for the case of more errors are summarized.

6. **Software**
   Software created for purposes of this thesis is described. Source code can be found on the enclosed CD.

CHAPTER 1

Preliminaries

In this chapter, we describe our motivation to study lossless seeds and mention some common applications. We describe a structure called free monoid, which is the basic tool for handling strings mathematically. Then we provide a rigorous definition of lossless seeds and terms connected to them. Finally, we show how lossless seeds can be used for mapping to a reference genome.

Molecular biology is one of the fastest growing fields of science. For instance, methods of DNA sequencing have been greatly improved throughout the last years. One of the results of this rapid development are huge nucleotide and protein sequence databases. Searching in these huge databases is still a very challenging problem. Well-known and very efficient string matching algorithms developed mainly for searching in text databases are too slow when applied to larger databases, therefore, computer scientists are currently looking for new methods and algorithms. One of these methods that has already proven its value uses a so-called seeding technique.

Seeding is based on the simple idea that if two strings of the same length are similar, i.e., they differ only in few characters, they must share an exact pattern, the so-called seed. A seed is a string consisting of joker symbols `-` corresponding to either a match or a mismatch, and match symbols `#`.

For example, if two strings of length 15 match within two errors, the shared patterns are, e.g., `#-##--#-##` and `#####`. The second one is the longest possible contiguous seed in this case. This also demonstrates the advantage of spaced seeds in comparison with contiguous seeds; for the same task, we can find spaced seeds of higher weight (weight is the number of occurrences of `#` in the seed).

There are two basic applications of seeds:

i) **sequences alignment**, where we detect potential similarity regions, i.e., those sharing the patterns; then we check whether the corresponding alignment meets our requirements;

ii) **approximate string matching**, which is described bellow.

Since classical algorithms for text searching within few errors are slow, filtering is a widely used approach how to improve efficiency. It consists of two phases:

1. the *filtration phase*: application of a filter which very quickly excludes highly dissimilar pairs of strings;

2. the *verification phase*: slower checking whether the few pairs passing the filtration phase are really within the given number of errors.

Efficiency of a filter is measured by two parameters:

- *sensitivity*: measures ratio of false negatives (i.e, pairs filtered out despite of they should be passed);

- *selectivity* (*specificity*): measures ratio of false positives (i.e, pairs which passed the filter despite of they should be filtered out), aim is to minimize it.

In the context of lossless search, we require sensitivity to be maximal, i.e., we do not allow any false negatives, therefore, only the selectivity parameter matters. Let us remark that lossy search seems more interesting from the practical viewpoint since a small diminish of the sensitivity can significantly improve the selectivity. Nevertheless, in several applications we require lossless search and this case is studied in this thesis.

We suppose that letters in strings are i.i.d., hence we consider the Bernoulli model of matching, i.e., probability of a single character match is p and of a mismatch $1 - p$. It is readily seen that higher weight of a seed implicates also better selectivity.

Combinatorial design of lossless seeds with high weights remains a challenging problem. In this work, we propose new methods for it.

1.1 Summary of articles

Let us briefly summarize basic papers in the field of lossless spaced seeds. In this thesis we study only the case of a single seed and one hit.

This framework is proposed for the first time in [1]. Many new ideas for lossless spaced seed design and the first results concerning the asymptotic properties of seeds appear in [3]. Hardness of optimal spaced seed design is studied in [6]. An exhaustive study of asymptotic behavior of optimal seeds is provided in [5]. An example of practical usage of lossless seeds can be found in [7]. Usage of mathematical objects called perfect rulers to construct seeds is presented in [8].

For completeness, let us mention that the case of more hits of a single seed is studied in [1] and the generalization to the case of (disjunctive) multi-seed families and more hits is presented in [3]. A very recent paper [9] introduces an approach for multi-seed families design based on the number theory.

The idea of lossy seeds appears in the PatternHunter program ([2]) for the first time and it is used in several other popular programs (e.g., [4]).

A list of all articles about (both lossy and lossless) spaced seeds is regularly maintained in [10]. Note that the number of these papers has already exceeded one hundred.

1.2 Free monoid $(\mathcal{A}^*, \cdot)$

An *alphabet* is a finite non-empty set $\mathcal{A} = \{a_0, a_1, \dots, a_{d-1}\}$. Its elements are called *letters*. A *string* S is a finite sequence over $\mathcal{A}$. The number of letters in the string S is called *length* of S and we denote it by $|S|$. A special case is the unique string of zero

length called *empty string* and denoted by ε. Let us denote the number of occurrences of a letter $a \in \mathcal{A}$ in the string S by $|S|_a$.

$\mathcal{A}^*$ is the set of all strings over $\mathcal{A}$ and $\mathcal{A}^+$ is the set of all strings over $\mathcal{A}$ except ε. $\mathcal{A}^n$ is the subset of $\mathcal{A}^*$ containing all strings of length n. $\mathcal{A}^{\mathbb{N}}$ is the set of all *infinite strings*, i.e., infinite sequences over $\mathcal{A}$.

We can equip the sets $\mathcal{A}^*$ and $\mathcal{A}^+$ with a binary operation called *concatenation*: if $S = S_0S_1 \cdots S_{m-1}$ and $T = T_0T_1 \cdots T_{n-1}$, where $S_i, T_i \in \mathcal{A}$, then

$$S \cdot T = S_0S_1 \cdots S_{m-1}T_0T_1 \cdots T_{n-1}.$$

If there cannot be any misunderstanding, the symbol $\cdot$ is omitted.

We obtain:

- the *free semigroup* $(\mathcal{A}^+, \cdot)$;
- the *free monoid* $(\mathcal{A}^*, \cdot)$ with the neutral element ε.

A string U is a *substring* of a (possibly infinite) string S if $S = TUV$ for some $T \in \mathcal{A}^*$ and $V \in \mathcal{A}^* \cup \mathcal{A}^N$. In the case $T = \varepsilon$, we say that U is a *prefix* of S. Similarly, if $V = \varepsilon$, then U is a *suffix* of S. By $S^\omega = SSS \cdots$ we denote repetition of the string S infinitely many times.

1.3 Similarities and lossless seeds

Now we define rigorously lossless seeds, similarities and (m, k)-problems.

Definition 1. A string over the alphabet $\{\texttt{\#}, \texttt{-}\}$ is called a *seed*. The letter `#` is called the *solid position symbol* and the letter `-` is called the *joker symbol*.

Remark 1. In many papers, there is used a different definition of seeds. Authors often require that every seed Q must also satisfy that the first and the last symbol is `#`.

Definition 2. For a seed Q we define two basic functions:

- *span* of the seed Q as its length, i.e.,
$$s(Q) := |Q|;$$
- and *weight* of the seed Q as the number of occurrences of the solid position symbols in the seed, i.e.,
$$w(Q) := |Q|_{\texttt{\#}}.$$

Definition 3. A *similarity* is a string over the alphabet $\{0, 1\}$. A similarity L of length m such that$|L|_0 = k$ is said to be (m, k)*-similarity*.

For a given m and pairwise distinct indices $0 \leq i_0, i_1, \ldots, i_{k-1} < m$, we use for (m, k)-similarities the following notation:

$$\left(L[i_0, \ldots, i_{k-1}]\right)_j = \begin{cases} 0 & \text{if } j \in \{i_0, i_1, \ldots, i_{k-1}\}, \\ 1 & \text{otherwise.} \end{cases}$$

Definition 4. Consider two strings S and T of length m. We define *similarity of strings S and T* as a string $L(S,T)$ of length m, which satisfies

$$\begin{aligned} (L(S,T))_i = 1 \quad &\Leftrightarrow \quad S_i = T_i \\ (L(S,T))_i = 0 \quad &\Leftrightarrow \quad S_i \neq T_i \end{aligned}$$

for all $i \in \{0, 1, \ldots, m-1\}$.

Definition 5. Consider a seed Q of span s and an (m,k)-similarity L such that $s \leq m$. We say that the seed Q *detects the similarity* L at position $i \in \{0, 1, \ldots, m-s\}$ if for all $j \in \{0, 1, \ldots, s-1\}$, it holds

$$Q_j = \texttt{\#} \quad \Rightarrow \quad L_{i+j} = 1.$$

Example 1. A similarity 0110110111101 is detected by the seed `#-#--#-#` at positions 2 and 5.

```
0 1 1 0 1 1 0 1 1 1 1 0 1
    # - # - - # - #
          # - # - - # - #
```

Definition 6. A seed Q is said to *solve an (m,k)-problem* for some m and k if it detects the all possible (m,k)-similarities.

Remark 2. The number of all possible (m,k)-similarities is $\binom{m}{k}$.

Observation 1. If a seed Q solves an (m,k)-problem for some m, than it solves also every (m',k)-problem for all $m' > m$.

Definition 7. Given an (m,k)-problem. A seed Q of weight w and span s such that $s < m$ is called *optimal*, if it solves the (m,k)-problem and all other seeds solving the (m,k)-problem have lesser or equal weight.

In stringology various kinds of distances (let us say between some strings S and T) are defined. They are usually based on the minimal number of steps necessary to transform S to T. The most common distances are:

- the Hamming distance, where the step is a replacement of an individual character;
- the Levenshtein distance, where the step is either a replacement, an insertion, or a deletion of an individual character.

In this text, we consider only the Hamming distance.

Definition 8. For two strings S and T of equal length, the *Hamming distance* $d_{\mathrm{Ham}}(S,T)$ is defined as the number of differing positions, i.e.,

$$d_{\mathrm{Ham}}(S,T) = \#\{i \mid S_i \neq T_i\}.$$

Now we can connect the solvability of (m,k)-problems with the similarity search.

Proposition 1. A seed Q solves the (m,k)-problem if and only if it detects a similarity $L(S,T)$ of any two strings S and T of length m with $d_{\mathrm{Ham}}(S,T) \leq k$.

Proof. The seed Q solves the (m,k)-problem $\Leftrightarrow$ Q detects all (m,k)-similarities $\Leftrightarrow$ Q detects all (m,k')-similarities for all $k' \leq k \Leftrightarrow Q$ detects $L(S,T)$ for any strings S and T of length m such that $d_{\mathrm{Ham}}(S,T) \leq k$. □

Example 2. The $(4,1)$-problem is solved, e.g., by the seed $Q =$ `#-#` as follows from examining of all $L[i]$ similarities:

string S :	**c**	c	t	g
string T :	**a**	c	t	g
similarity $L[0]$:	**0**	1	1	1
seed Q :		`#`	`-`	`#`

string S :	c	**c**	t	g
string T :	c	**t**	t	g
similarity $L[1]$:	1	**0**	1	1
seed Q :	`#`	`-`	`#`	

string S :	c	c	**t**	g
string T :	c	c	**a**	g
similarity $L[2]$:	1	1	**0**	1
seed Q :		`#`	`-`	`#`

string S :	c	c	t	**g**
string T :	c	c	t	**a**
similarity $L[3]$:	1	1	1	**0**
seed Q :	`#`	`-`	`#`	

To decide whether a seed Q solves an (m,k)-problem, where $0 < k < m$, is an NP-hard task, hence the worst case complexity is at least exponential in the size of input, which is $\Theta(s(Q)+m)$ or $\Theta(s(Q)+\log(m))$ in the case of unary or binary notation, respectively.

Theorem 1 ([6, problem of non-detection])**.** The problem to decide whether a seed Q solves an (m,k)-problem is strongly NP-complete, i.e., NP-complete even when integers are represented in the unary notation.

To complete the list of basic terms, we mention a definition of the so-called cyclic (m,k)-problems.

Definition 9. We say that a seed Q of span m solves the *cyclic* (m,k)*-problem* if for every (m,k)-similarity L, there exists some $i \in \{0,1,\ldots,m-1\}$ such that $\sigma_i(Q)$ detects L, where

$$\sigma_i : \{\texttt{-},\texttt{\#}\}^n \to \{\texttt{-},\texttt{\#}\}^n$$

denotes the i-th *cyclic shift*, i.e.,

$$\sigma_i(t_0 t_1 \cdots t_{n-1}) = t_i t_{i+1} \ldots t_{n-1} t_0 \cdots t_{i-1},$$

for t_j from the alphabet $\{\texttt{-},\texttt{\#}\}$.

Example 3. The seed $Q =$ `####-#---` solves cyclically the $(9,2)$-problem. For every $(9,2)$-similarity, consider, for instance, $L = 011101111$, we can find an appropriate cyclic shift of Q.

0	1	1	1	0	1	1	1	1
`-`	`#`	`-`	`-`	`-`	`#`	`#`	`#`	`#`

1.4 Application of lossless seeds for mapping to a reference genome

To conclude this chapter, let us mention one of the basic applications of lossless seeds — the similarity search in DNA.

Consider that we have a reference genome G of length n over the alphabet $\mathcal{A}_{\mathrm{DNA}} = \{c, t, a, g\}$. Consider the following task: for every $S \in \{S_0, S_1, \ldots, S_{r-1}\}$ of words from $\mathcal{A}^m_{\mathrm{DNA}}$, where $m << n$, find all positions p such that

$$d_{\mathrm{Ham}}(G_p G_{p+1} \cdots G_{p+m-1}, S) \leq k \tag{1.1}$$

for some fixed k. In other words, for all S_i find similar substrings of G.

In our approach, we use a seed solving the (m, k)-problem (if it is possible, the optimal one), let us say Q, of span s and weight w. Then we define a map $\Upsilon : \mathcal{A}^s_{\mathrm{DNA}} \to \mathcal{A}^w_{\mathrm{DNA}}$ as

$$\Upsilon(W) = W_{i_0} W_{i_1} \cdots W_{i_{w-1}},$$

where $i_0 < i_1 < \cdots < i_{w-1}$ are indices of `#`'s in the seed.

We create an index of G based on a hash table. We iterate through G, take all its substrings F of length s and store the position (let us say i) of the substring F in the genome G to the line $\Upsilon(F)$ of the table.

To perform the similarity search of a string S in the genome G, we take all substrings V of S of length s. For each V (in S is at position j), we look into the hashtable on the line $\Upsilon(F)$ and pick up all the indices i. Then we check whether the substring of G of length m corresponding to the indices i, j is in the required Hamming distance to S, i.e., if equation (1.1) holds for $p = i - j$. We demonstrate the whole procedure on the following example.

Example 4. Consider the task of searching strings of length 11 up to 2 errors in the string

$$G = ggtagaggctaggccaag$$

of length 18.

1. **Index building**

   For this purpose, we take an optimal seed $Q =$ `##---##` and create the hash table.

position i	substring F	$\Upsilon(F)$
0	*ggtagag*	*ggag*
1	*gtagagg*	*gtgg*
2	*tagaggc*	*tagc*
3	*agaggct*	*agct*
4	*gaggcta*	*gata*
5	*aggctag*	*agag*
6	*ggctagg*	*gggg*
7	*gctaggc*	*gcge*
8	*ctaggcc*	*ctec*
9	*taggcca*	*taca*
10	*aggccaa*	*agaa*
11	*ggccaag*	*ggag*

$\Rightarrow$

$\Upsilon(F)$	positions i
agaa	10
agag	5
agct	3
ctec	8
gata	4
gcge	7
ggag	0, 11
gggg	6
gtgg	1
taca	9
tagc	2

2. **Searching**

   (a) **A successful search**

   Now we want to find occurrence of a string similar to the string

$$S = gacgctagacc$$

   of length 11.

position j	substring V	$\Upsilon(V)$
0	*gacgcta*	*gata*
1	*acgctag*	*acag*
2	*cgctaga*	*cgga*
3	*gctagac*	*gcac*
4	*ctagacc*	*ctcc*

   The only candidate is

$$j = 0,\ gata \quad \sim \quad gata,\ i = 4;$$

   which corresponds to the substring

$$G_4G_5 \cdots G_{14} = gaggctaggcc.$$

   The final step is verification that the Hamming distance of both strings is less or equal to 2. In this case, it is really 2.

$S =$	g	a	c	g	c	t	a	g	a	c	c
$G_4 \cdots G_{14} =$	g	a	g	g	c	t	a	g	g	c	c
	1	1	0	1	1	1	1	1	0	1	1

   Hence we have found a substring of G similar to S:

$$ggta\mathbf{gaggctaggcc}aag.$$

   (b) **An unsuccessful search**

   Now consider the string

$$S = gttgatggtag$$

   of length 11.

position j	substring V	$\Upsilon(V)$
0	*gttgatg*	*gttg*
1	*ttgatgg*	*ttgg*
2	*tgatggt*	*tggt*
3	*gatggta*	*gata*
4	*atggtag*	*atag*

   Again, we have only one candidate:

$$j = 3,\ gata \quad \sim \quad gata,\ i = 4;$$

which corresponds to the the substring

$$G_1 G_2 \cdots G_{11} = gtagaggctag.$$

Nevertheless, the Hamming distance of the strings is 3.

$S =$	g	t	t	g	a	t	g	g	t	a	g
$G_1 \cdots G_{11} =$	g	t	a	g	a	g	g	c	t	a	g
	1	1	0	1	1	0	1	0	1	1	1

Thus we have not found a substring of G similar to S.

CHAPTER 2

Lossless seeds survey

This chapter summarizes published results concerning lossless seeds for the case of single seed and one hit. First some simple transformational properties of seeds are mentioned. Afterwards, we recall asymptotic properties of seeds. The last paragraphs are devoted to usage of perfect rulers for $(m, 2)$-seed design.

We begin with this remark: good lossless seeds and lossy seeds show different features in some sense. While lossy seeds are usually very random with no visible structure, many of high quality lossless seeds seem to be quite regular.

2.1 Some basic properties

Let us present some general properties of seeds as described in [3].

Lemma 1. If a seed Q of span s solves an (m, k)-problem and $\ell = m - s$, then every substring T of Q of length t solves the $(t + \ell, k)$-problem.

Transformation between certain (m, k)-problems can be performed using the regular expansion and the regular contraction.

Definition 10. Let Q be a seed of span s. Its *i-th regular expansion* is a seed $P = i \otimes Q$ of span $i(s-1)+1$ such that

$$P_j = \begin{cases} Q_{\frac{j}{i}} & \text{for all } j \in \{0, i, 2i, \ldots, i(s-1)\}, \\ \texttt{-} & \text{otherwise.} \end{cases}$$

Example 5 (Regular expansion of the seed $Q =$ `##-#-###`).

$$\begin{aligned} 1 \otimes Q &= \texttt{\#\#-\#-\#\#\#} \\ 2 \otimes Q &= \texttt{\#-\#---\#---\#-\#-\#} \\ 3 \otimes Q &= \texttt{\#--\#-----\#-----\#--\#--\#} \\ 4 \otimes Q &= \texttt{\#---\#-------\#-------\#---\#---\#} \\ &\vdots \end{aligned}$$

Lemma 2. 1. If a seed Q solves an (m, k)-problem, then the $(im, (i+1)k-1)$-problem is solved both by the seed Q and its i-regular expansion $i \otimes Q$.

2. If a seed $Q^{(i)} = i \otimes Q$ solves an (im, k)-problem, then Q solves both the (im, k)-problem and the $(m, \lfloor \frac{k}{i} \rfloor)$-problem.

2.2 Asymptotic properties

Asymptotic properties of seeds are studied in [3] and [5]. In what follows, we select the most important results.

The first paper provides only the asymptotic bound on w when k is fixed and m goes to infinity, whereas the second gives powerful theorems with various asymptotic relations. Some of the variables (such as m, k, the number of jokers, w) are fixed and it is studied what the other variables must satisfy.

First let us shortly recall the asymptotic notation used in this section. Consider two functions $f(x_1, \dots, x_n)$ and $g(x_1, \dots, x_n)$ defined on some subset of $(\mathbb{R}_0^+)^n$. Then we say that:

- $f(x_1, \dots, x_n) \in \mathcal{O}(g(x_1, \dots, x_n))$ if there exist positive constants c_0 and c such that
$$\forall i \in \{1, \dots, n\}(x_i > c_0) \quad \Longrightarrow \quad f(x_1, \dots, x_n) \leq cg(x_1, \dots, x_n);$$
- $f(x_1, \dots, x_n) \in \Omega(g(x_1, \dots, x_n))$ if there exist positive constants c_0 and c such that
$$\forall i \in \{1, \dots, n\}(x_i > c_0) \quad \Longrightarrow \quad f(x_1, \dots, x_n) \geq cg(x_1, \dots, x_n);$$
- $f(x_1, \dots, x_n) \in \Theta(g(x_1, \dots, x_n))$ if both the following conditions are satisfied:
$$f(x_1, \dots, x_n) \in \mathcal{O}(g(x_1, \dots, x_n)),$$
$$f(x_1, \dots, x_n) \in \Omega(g(x_1, \dots, x_n)).$$

2.2.1 Maximization of the weight when the number of errors is fixed

We begin with the paper [3], which has initialized study of asymptotic properties. The authors describe the asymptotic behavior of the optimal seeds in the case of cyclic (m, k)-problems with fixed k and then modify it to (m, k)-problems.

Lemma 3. For a fixed positive k, let $w(m)$ denote the maximal weight of the seeds solving the cyclic (m, k)-problem in dependence on m. Then the following holds:

$$m - w(m) \in \Theta\left(m^{\frac{k-1}{k}}\right).$$

Remark 3. In the proof of this lemma, an explicit construction of asymptotically optimal seeds is presented. As it is an important result, we recall it here.

Let us consider the cyclic (m, k)-problem for given m and k. Let us denote $B = \lceil m^{\frac{1}{k}} \rceil$. Then we construct the following seeds, from which we create the final seed Q.

$$\begin{aligned}
Q^{(0)} &= \text{ prefix of } \left(-\#^{B-1}\right)^{\omega} && \text{of length } m,\\
Q^{(1)} &= \text{ prefix of } \left(-^{B}\#^{B(B-1)}\right)^{\omega} && \text{of length } m,\\
&\vdots\\
Q^{(i)} &= \text{ prefix of } \Big(\underbrace{-^{B^i}\#^{B^i(B-1)}}_{\text{length } B^{i+1}}\Big)^{\omega} && \text{of length } m,\\
&\vdots\\
Q^{(k-1)} &= \text{ prefix of } \left(-^{B^{k-1}}\#^{B^{k-1}(B-1)}\right)^{\omega} && \text{of length } m.
\end{aligned} \tag{2.1}$$

The desired final seed Q is obtained from the seeds $Q^{(0)}, \ldots, Q^{(k-1)}$ as a "`#` conjunction" ("`-` disjunction") of these seeds, i.e.:

$$\forall i \in \{0, 1, \ldots, m-1\}: \qquad Q_i = \begin{cases} \texttt{\#} & \text{if } \bigwedge_{j \in \{0,1,\ldots,k-1\}} \left[(Q^{(j)})_i = \texttt{\#} \right], \\ \texttt{-} & \text{otherwise.} \end{cases} \tag{2.2}$$

Example 6. Consider the cyclic $(40, 2)$-problem. Then $B = 7$.

$Q^{(0)}$	`=-######-######-######-######-######-####`
$Q^{(1)}$	`=-------##################################`
Q	`=--------######-######-######-######-####`

Example 7. Consider the cyclic $(40, 3)$-problem. Then $B = 4$.

$Q^{(0)}$	`=-###-###-###-###-###-###-###-###-###-###`
$Q^{(1)}$	`=----############----############----####`
$Q^{(2)}$	`=----------------######################`
Q	`=--------------------###-###-###-----###`

From the properties of cyclic (m, k)-problems follow the properties of (m, k)-problems. One can transform the tight bound on optimal seeds given in Lemma 3 to a tight bound on $m - w(m)$ for (m, k)-problems, when k is fixed, as it is stated in the following proposition.

Proposition 2. For a fixed positive k, let $w(m)$ denote the maximal weight of the seeds solving the (m, k)-problem in dependence on m. Then the following holds:

$$m - w(m) \in \Theta\left(m^{\frac{k}{k+1}}\right).$$

In the proof, the authors use Lemma 3 in the following way. For given m and k, they find appropriate constants p and s. Then they construct a seed solving the cyclic (p, k)-problem and find a substring of P^ω of length s of the highest possible weight (for more details on pattern repeating, see Section 2.3, in particular, Lemma 4). Finally they prove that for a fixed k, one obtains the mentioned bound.

As a corollary of the proposition, we obtain a simple criterion that can be used to show that a given algorithm does not provide asymptotically optimal seeds.

Corollary 1. Let k be fixed. Then for both cyclic (m, k)-problems and (m, k)-problems, it holds

$$\lim_{m \to +\infty} \frac{w(m)}{m} = 1. \tag{2.3}$$

2.2.2 Maximization of the weight when the number of errors varies

Stronger results on asymptotic properties of seeds are obtained in [5]. First we recall two theorems, which concern the function $w(m, k)$.

Theorem 2. Let $w(m, k)$ denote the maximal possible weight of the seeds solving the (m, k)-problem. For every $k < \frac{1}{8} \log_2 m$, it holds that:

$$m - w(m, k) \in \begin{cases} \mathcal{O}\left(k m^{\frac{k}{k+1}}\right) & \text{if } k < \log\log m, \\ \mathcal{O}\left(m^{\frac{k}{k+1}}\right) & \text{if } k \geq \log\log m, \end{cases}$$
$$m - w(m, k) \in \Omega\left(m^{\frac{k}{k+1}}\right).$$

Note that Proposition 2 can be obtained from Theorem 2 easily.

Remark 4. Again, there appear an explicit seed construction in the proof. The idea is almost the same as in the method described in Remark 3.

Let us consider the (m,k)-problem for given m and k. Let us assign $\tilde{B} = \left\lfloor m^{\frac{1}{k+1}} \right\rfloor$ and $s = m - 2\sum_{i=1}^{k} B^i$.

Then let us compute seeds $Q^{(0)}, Q^{(1)}, \ldots, Q^{(k-1)}$ using the formulae (2.1) with two small changes — we replace B by $\tilde{B}$, and m by s. Hence the produced seeds are of span s. The final seed Q is obtained from $Q^{(0)}, Q^{(1)}, \ldots, Q^{(k-1)}$ again by (2.2).

Example 8. Consider the $(48,2)$-problem. Then $B = 3$ and $s = 24$.

$Q^{(0)}$	=	`-##-##-##-##-##-##-##-##`
$Q^{(1)}$	=	`---######---######---###`
Q	=	`----##-##----##-##----##`

Theorem 3. Let $w(m,k)$ denote the maximal possible weight of the seeds solving the (m,k)-problem. For every $k \geq \frac{1}{8}\log_2 m$, it holds that:

$$w(m,k) \in \mathcal{O}\left(\frac{m}{k}\log m\right),$$
$$w(m,k) \in \Omega\left(\frac{m}{k}\log\frac{m}{k}\right).$$

2.2.3 Maximization of the number of errors

Now let us consider the task when the number of jokers, length of the seeds, and m are given, and the question is how many errors can be detected using some seed at most.

Proposition 3. Let the number of jokers g be fixed. Let $k(m,s,g)$ denote the maximal number of errors k such that there exists a seed Q of span s and $s - w(Q) = g$ solving the (m,k)-problem.

Then

$$\left| k(m,s,g) - \frac{2g+1}{g+1}\cdot\frac{m}{s} \right| \in \mathcal{O}\left(\max\left(1, \frac{m}{s^2}\right)\right).$$

Proposition 4. Let $k(m,s,g)$ denote the maximal number of errors k such that there exists a seed Q of span s satisfying $s - w(Q) = g$ and solving the (m,k)-problem. For every integer $r > 1$, if $s - w(Q) \leq s^{\frac{r-1}{r}}$, then

$$k(m,s,g) - r\cdot\frac{m}{s} \in \mathcal{O}(1).$$

2.3 Periodic seeds

Equipped with the results on the asymptotic properties of seeds, let us ask the following question. Can be the asymptotic bound on optimal seeds reached by repeating a fixed pattern?

First recall under which circumstances such repeating is possible. The following lemma exactly corresponds to the first part of Lemma 3 in [3].

Lemma 4. If $\ell > 0$ and a seed P of span $p = \ell + 1$ solves the cyclic (p,k)-problem, then every substring Q of P^ω of span s solves the $(s+\ell,k)$-problem.

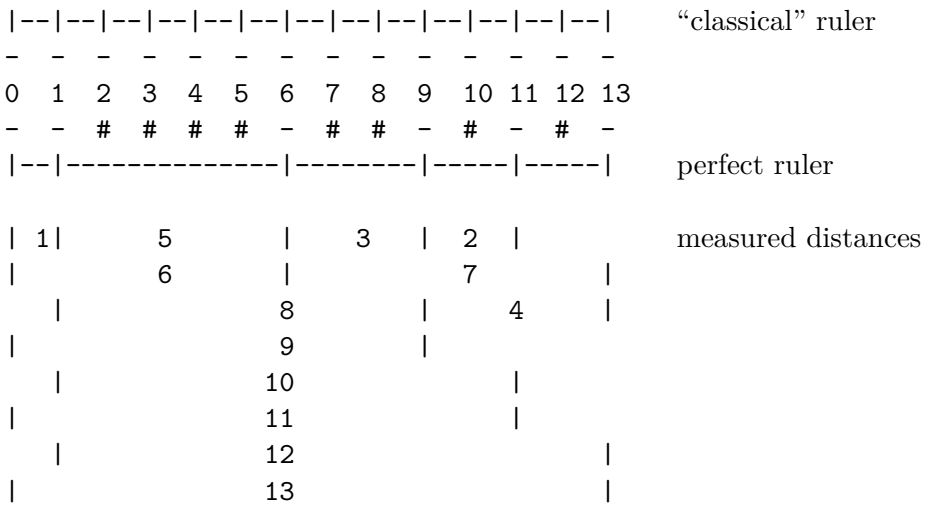


Figure 2.1: A perfect ruler and a "classical ruler", both of span 14 (thus they measure distances from $\{1, 2, \ldots, 13\}$).

As follows from Corollary 1, the answer to the question is negative since, for optimal seeds, $\frac{\text{the number of occurrences of \#}}{m}$ must tend to 1. Nevertheless, in the case of a repeated pattern P, it tends to $\frac{w(P)}{s(p)} < 1$. However, one can change and extend the repeated pattern with growing m. This approach is studied in Section 4.3.

The last lemma is presented just for completeness and, again, it follows from Lemma 3 in [3]. It gives us a tool how to verify the cyclic (m, k)-solvability of a seed.

Lemma 5. Consider a seed P of span p such that PP solves a $(3p - 1, k)$-problem for some k. Then P solves the cyclic (m, k)-problem.

2.4 Perfect rulers

An interesting method of seed design based on the so-called perfect rulers is presented in [8]. The authors suggest a straightforward way how to extend perfect rulers to seeds solving $(m, 2)$-problems. Pairs of errors in shorter distances are detected inside the seed, whereas for the cases of pairs of errors with longer distances, the seed is moved to the left or the right.

Definition 11. A seed Q over $\{\texttt{\#}, \texttt{-}\}$ of span s is said to be *complete ruler* if every other complete ruler of the same span has lesser or equal weight.

Remark 5. If a seed Q of span s is a complete ruler, then $Q_0 = Q_{s-1} = \texttt{-}$.

Definition 12. A complete ruler R of span r is called a *perfect ruler* if there is no complete ruler of the same span with higher weight. We denote $\ell(d) := d + 1 - w(R)$, where R is an arbitrary perfect ruler of span $d + 1$.

Example 9. Rulers as defined above have a simple analogy to rulers from everyday life. We demonstrate a perfect ruler of span 14, `--####-##-#-#-`, and a classical one in Figure 2.1. If we use `-`'s for markers, we can measure all integer distances from 1 to 13.

Remark 6. Perfect rulers (known also as difference bases) were studied in deep in the field of discrete mathematics. It was, e.g., proven that

$$\lim_{d\to+\infty} \frac{\ell^2(d)}{d} \in [2.424, 3.000].$$

In general, perfect rulers are difficult to find with an exception of Wichmann rulers, the complete rulers $W_{r,q}$ of the form of

$$W_{r,q} = {-}^{r+1}\#^{r}{-}(\#^{2r}{-})^{r}(\#^{4r+2}{-})^{q}(\#^{2r+1}{-})^{r+1}{-}^{r}, \tag{2.4}$$

which are famous for being perfect. For a Wichmann rulers $W_{r,q}$, it holds

$$\begin{aligned} s(W_{r,q}) &= 4(r+1)^2 + q(4r+3), \\ s(W_{r,q}) - w(W_{r,q}) &= 4r + q + 3. \end{aligned}$$

More details can be found in [11]

When having a perfect ruler R, the authors suggest adding #'s to the beginning and the end of R in order to obtain a better seed Q. Then they find out for which minimal m^*_Q the seed Q solves the $(m^*_Q, 2)$-problem.

The following theorem provides basic estimates on m^*_Q. The upper one holds for wrapped complete rulers, whereas the lowers hold for arbitrary wrapped seeds with no restrictions. The estimates imply that, from practical point of view, it does not make any sense to take $s_0 > r-1$ and $s_1 > r-1$ since then m would be required to be too big.

Theorem 4. Let

$$Q = \#^{s_0} R \#^{s_1}, \tag{2.5}$$

where R is a seed of span r.

1. If R is a complete ruler and $\max(s_0, s_1) \le r-1$, then

$$m^*_Q \le 2 \cdot s(Q) - 1 - \min(s_0, s_1).$$

2. If $\max(s_0, s_1) > r-1$, then

$$m^*_Q \ge 2 \cdot s(Q).$$

3. If $\min(s_0, s_1) > r-1$, then

$$m^*_Q \ge 2 \cdot s(Q) + \min(s_0, s_1).$$

This theorem can be used for $(m, 2)$-seeds construction. For a given m, we consider seeds Q of the form of (2.5) with perfect rulers R of spans r satisfying

$$\max(s_0, s_1) \le r-1 \qquad \text{and} \qquad s(Q) \le \frac{m + \min(s_0, s_1) + 1}{2},$$

from whose we choose some of those with the highest weight.

2.5 Summary

Lossless seed design still remains an unsolved task. The mentioned methods do not provide satisfying algorithms for generating seeds of enough weight. The asymptotic properties of seeds are known very well and the statements provide an insight on how the optimal seeds must look like, and immediately exclude, e.g., repeating a fixed pattern.

The idea of perfect rulers seems to be interesting from theoretical point of view. It enables to design seeds from mathematical structures which have been studied in other branches of mathematics. However, experiments show that the obtained seeds have lower weights than those constructed, e.g., using the greedy algorithm introduced in Section 4.2. Moreover, the methods based on greedy algorithm have much better time complexity than methods based on perfect rulers, since the perfect ruler design is known to be NP-hard. On the other hand, the method could be significantly improved if we connected some more complicated left and right strings instead of the simplest strings of the form of `#`$\cdots$`#`, which are the only connected strings studied in the original paper.

The idea of rulers appears also in the following chapters, nevertheless, cyclic rulers (defined in Section 4.3) seem to be a better alternative for $(m, 2)$-seed design than perfect rulers are.

CHAPTER 3

One error

In this chapter, we study systematically the $(m, 1)$-problem, which is simple to handle. However, to the best of our knowledge, the necessary and sufficient criterion presented here has never appeared in any paper. Note that only some properties of $(m, 1)$-seeds are mentioned in [3].

Let us fix m and consider the $(m, 1)$-problem. Let us have a seed Q of span s. We denote $\ell = m - s$, hence Q can be located at $\ell + 1$ different positions. If we consider all possible $L[i]$ similarities, we obtain the following criterion.

Theorem 5. Let s and m be positive integers such that $m \geq s$ and Q be a seed of span s. Denote $\ell = m - s$. Then the seed Q solves the $(m, 1)$-problem if and only if it does not contain $\texttt{\#}^{\ell+1}$ as a substring.

Proof. $\Rightarrow$: For a contradiction suppose that it contains $\texttt{\#}^{\ell+1}$ and it solves the problem, i.e., for all $j \in \{0, 1, \ldots, s-1\}$ the similarity $L[j]$ is detected. Find an index $i \in \{0, 1, \ldots, s-\ell-1\}$ such that $Q_i Q_{i+1} \cdots Q_{i+\ell} = \texttt{\#}^{\ell+1}$. Then the similarity $L[i+\ell]$ cannot be detected by Q, which is a contradiction.

$\Leftarrow$: Consider a similarity $L[i]$, where $i \in \{0, 1, \ldots, s-1\}$. If $i \in \{0, 1, \ldots, \ell-1\}$ or $i \in \{s-\ell, s-\ell+1, \ldots, s-1\}$, then $L[i]$ is detected by Q at position $\ell + 1$ or 0, respectively. Otherwise, there must necessarily exist $j \in \{0, 1, \ldots, \ell\}$ such that $S_{i-j} = \texttt{-}$, hence the seed Q detects the similarity $L[i]$ at position j. □

Example 10. Consider the $(10, 1)$-problem and seeds $Q^{(1)} = \texttt{\#-\#-\#\#-\#}$ and $Q^{(2)} = \texttt{\#-\#-\#\#\#\#}$ of span 8. As follows from Figure 3.1, the seed $Q^{(1)}$ solves the problem, whereas $Q^{(2)}$ does not since it contains `###`.

It is readily seen from Theorem 5 that, for given m and s, some of the best seeds can be designed using greedy algorithm – we place the symbol `#` ℓ-times, then we put the symbol `-` once. These steps are repeated until we reach the end of the seed.

An optimized version of this procedure appears in Algorithm 1. From all candidates proposed by the procedure, one for each $s \in \{2, 3, \ldots, m-1\}$, the best one (having the highest weight) is chosen. The algorithms finds an optimal seed solving the $(m, 1)$-problem for a given m. It has both time and space complexity $\mathcal{O}(m)$. Remark that the number of occurrences of `-` in the seed for a specific s is $\lfloor \frac{s}{m-s+1} \rfloor$.

$Q^{(1)}$:

#	-	#	-	#	#	-	#			
	#	-	#	-	#	#	-	#		
		#	-	#	-	#	#	-	#	
			#	-	#	-	#	#	-	#
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

$Q^{(2)}$:

#	-	#	-	#	#	#	#			
	#	-	#	-	#	#	#	#		
		#	-	#	-	#	#	#	#	
			#	-	#	-	#	#	#	#
✓	✓	✓	✓	✓	✓	✓	X	✓	✓	✓

Figure 3.1: Example of seeds solving and not solving the $(10, 1)$-problem.

s	ℓ	w	seed
1	9	1	`#`
2	8	2	`##`
3	7	3	`###`
4	6	4	`####`
5	5	5	`#####`
6	4	5	`####-#`
7	3	6	`###-###`
8	2	6	`##-##-##`
9	1	5	`#-#-#-#-#`

Table 3.1: Results of the greedy algorithm for the $(10, 1)$-problem. The seeds of the highest weight are highlighted.

```
function getHeaviestM1Seed(m):
    // variables for the best seed
    s̃ := 0;
    w̃ := 0;
    for s = 1, 2, ..., m − 1 do
        // weight of seed of span s
        w := s − ⌊s/(m−s+1)⌋;
        if w > w̃ then
            // we have found a better
               seed
            s̃ := s;
            w̃ := w;
        end
    end
    // s̃ − w̃ is the number of
       occurrences of - in the best
       seed
    return prefix of (#^{m−s̃}-)^{s̃−w̃} of
    length s;
```

Algorithm 1: Function returning the heaviest $(m, 1)$-seed

Example 11. Consider the $(10, 1)$-problem. Algorithm 1 returns the first seed, $Q =$ `###-##`, from the candidates with the highest weight (displayed in Table 3.1).

Optimal seeds for different $(m, 1)$-problems can be found in Table 3.2. Note that there are only those obtained by the greedy algorithm.

m	w	seeds
6	3	`###`
		`##-#`
		`#-#-#`
7	4	`##-##`
8	4	`####`
		`###-#`
		`##-##-`
		`#-#-#-#`
9	5	`###-##`
		`##-##-#`
10	6	`###-###`
		`##-##-##`
11	6	`####-##`
		`###-###-`
		`##-##-##-`
12	7	`####-###`
		`###-###-#`
		`##-##-##-#`
13	8	`####-####`
		`###-###-##`
		`##-##-##-##`
14	9	`###-###-###`
15	9	`#####-####`
		`####-####-#`
		`###-###-###-`
		`##-##-##-##-#`
16	10	`#####-#####`
		`####-####-##`
		`###-###-###-#`
		`##-##-##-##-##`
17	11	`####-####-###`
		`###-###-###-##`
18	12	`####-####-####`
		`###-###-###-###`

Table 3.2: Optimal seeds for $(m, 1)$-problems obtained by the greedy algorithm.

CHAPTER 4

Two errors

In this chapter, we introduce the so-called "Laser method" based on incidence matrices of seeds, which provides a necessary and sufficient condition for $(m, 2)$-problems solvability. Then we study properties of seeds obtained by the greedy algorithm, and propose designing seeds based on the so-called cyclic rulers. We conclude with a conjecture stating that all optimal seeds can be constructed from the cyclic rulers.

4.1 Laser method

Having a criterion for $(m, 1)$-seeds given by Theorem 5, which enables to find optimal $(m, 1)$-seeds quickly, we would welcome a similar criterion for $(m, 2)$-seeds.

The criterion for $(m, 1)$-seeds is very simple since we do not have to consider positions of errors appearing concurrently. For the case of two errors, our aim is to generalize Theorem 5 in the most straightforward way. It is natural that, since we have one dimension more, the criterion might be two-dimensional in some sense. To consider similarities $L[i, j]$ detected by a given seed, we define a so-called incidence matrix of the seed.

Before introducing the definition of the incidence matrix we need an auxiliary map ϕ

Definition 13. For a string W of length n over $\{\#, -\}$, we define a map

$$\phi : \{0,1\}^n \to \{0,1\}^{n \times n}$$

as

$$(\phi(W))_{i,j} = \begin{cases} 1 & \text{if } W_i = - \wedge W_j = -, \\ 0 & \text{otherwise,} \end{cases}$$

where the indices i and j are taken from $\{0, 1, \ldots, n-1\}$.

Definition 14. We define the *incidence matrix* of a seed Q of span s in the $(m, 2)$-problem as

$$M_Q = \phi\left(-^{\ell} Q -^{\ell}\right),$$

where $\ell = m - s$.

Example 12. Consider the $(10, 2)$-problem and the seed $Q =$ `#--#---#`. Then:

$$M_Q = \left(\begin{array}{cc|cccccccc|cc} 1&1&0&1&1&0&1&1&1&0&1&1\\ 1&1&0&1&1&0&1&1&1&0&1&1\\ \hline 0&0&0&0&0&0&0&0&0&0&0&0\\ 1&1&0&1&1&0&1&1&1&0&1&1\\ 1&1&0&1&1&0&1&1&1&0&1&1\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 1&1&0&1&1&0&1&1&1&0&1&1\\ 1&1&0&1&1&0&1&1&1&0&1&1\\ 1&1&0&1&1&0&1&1&1&0&1&1\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ \hline 1&1&0&1&1&0&1&1&1&0&1&1\\ 1&1&0&1&1&0&1&1&1&0&1&1 \end{array}\right).$$

The basic properties of incidence matrices are summarized in the following observation.

Observation 2. An incidence matrix M_Q satisfies $M_Q = \left(M_Q\right)^\top$ and

$$M_Q \begin{bmatrix} 0,\dots,\ell-1 \\ 0,\dots,\ell-1 \end{bmatrix} = M_Q \begin{bmatrix} 0,\dots,\ell-1 \\ m,\dots,m+\ell-1 \end{bmatrix} = \\ = M_Q \begin{bmatrix} m,\dots,m+\ell-1 \\ 0,\dots,\ell-1 \end{bmatrix} = M_Q \begin{bmatrix} m,\dots,m+\ell-1 \\ m,\dots,m+\ell-1 \end{bmatrix} = J_\ell,$$

where J_ℓ denotes the $\ell \times \ell$ all-ones matrix and

$$M_Q \begin{bmatrix} i_0, i_1, \dots, i_{r-1} \\ j_0, j_1, \dots, j_{r-1} \end{bmatrix} = \begin{pmatrix} (M_Q)_{i_0,j_0} & (M_Q)_{i_0,j_1} & \dots & (M_Q)_{i_0,j_{r-1}} \\ (M_Q)_{i_1,j_0} & (M_Q)_{i_1,j_1} & \dots & (M_Q)_{i_1,j_{r-1}} \\ \vdots & \vdots & \ddots & \vdots \\ (M_Q)_{i_{r-1},j_0} & (M_Q)_{i_{r-1},j_1} & \dots & (M_Q)_{i_{r-1},j_{r-1}} \end{pmatrix}.$$

The following proposition describes the relation between incidence matrices and $(m, 2)$-similarities detection.

Proposition 5. For any positive m, consider the $(m, 2)$-problem and a seed Q of span s. Denote $\ell = m - s$. Let $t \in \{0, 1, \dots, \ell\}$. Then the following statements are equivalent:

i) the seed Q detects the similarity $L[i, j]$ at position t;

ii) $\left(M_Q\right)_{i+\ell-t, j+\ell-t} = 1$.

Proof. $\Rightarrow$: The seed Q detects $L[i, j]$ at position t, therefore, it also detects $L[i]$ at t. There are three possible situations:

1) $i \in \{0, 1, \dots, t-1\}$;
2) $i \in \{t, t+1, \dots, t+s-1\}$ and $Q_{i-t} =$ `-`;
3) $i \in \{t+s, t+s+1, \dots, m-1\}$.

In all these cases, $\left(\text{-}^\ell Q \text{-}^\ell\right)_{i+\ell-t} = 1$ is satisfied. By the same reasoning, $\left(\text{-}^\ell Q \text{-}^\ell\right)_{j+\ell-t} = 1$. Altogether, we get

$$\left(M_Q\right)_{i+\ell-t, j+\ell-t} = 1.$$

$\Leftarrow$: It follows easily from the fact that Q detects the error $L[i]$ at position t if and only if $(\text{-}^t Q \text{-}^{\ell-t})_i = \text{-}$. □

Now we can introduce the necessary and sufficient condition for $(m, 2)$-problems solvability.

Theorem 6. Given a seed Q of span s and the $(m, 2)$-problem. Denote $\ell = m - s$. The following statements are equivalent:

i) the seed Q solves the $(m, 2)$-problem;

ii) the incidence matrix M_Q of the seed Q does not contain any $(\ell+1) \times (\ell+1)$ block of the form

$$\begin{pmatrix} 0 & \star & \dots & \star & \star \\ \star & 0 & \dots & \star & \star \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \star & \star & \dots & 0 & \star \\ \star & \star & \dots & \star & 0 \end{pmatrix}.$$

Formally: for all indices $i, j \in \{0, 1, \dots, m-1\}$, there exists an index $p \in \{0, 1, \dots, \ell\}$ such that $\big(M_Q\big)_{i+p, j+p} = 1$.

Proof. The idea of the proof is similar as in Theorem 5 in the previous chapter for the case of one error. Denote $M := M_Q$.

$\Rightarrow$: Let Q solve the problem and

$$M_{i,j} = M_{i+1,j+1} = \dots = M_{i+\ell,j+\ell} = 0$$

for some $i, j \in \{0, 1, \dots, m-1\}$. It follows from Proposition 5 that

$M_{i,j}$ corresponds to the detection of $L[i, j]$ at position ℓ,
$M_{i+1,j+1}$ corresponds to the detection of $L[i, j]$ at position $\ell - 1$,
$\vdots$ $\vdots$
$M_{i+\ell,j+\ell}$ corresponds to the detection of $L[i, j]$ at position 0.

Therefore,the similarity $L[i, j]$ remains undetected by Q. Thus Q does not solve the problem.

$\Leftarrow$: Consider a similarity $L[i, j]$, where $i, j \in \{0, 1, \dots, m-1\}$. By the correspondence mentioned above and because there must exist some $k \in \{0, 1, \dots, \ell\}$ such that $M_{i+k,j+k} = 1$, the similarity $L[i, j]$ is detected by the seed Q at position $\ell - k$. □

Corollary 2. If for some i, j and all $p \in \{0, 1, \dots, \ell\}$ we have

$$\big(M_Q\big)_{i+p, j+p} = 0,$$

then the seed Q does not detect the similarity $L[i, j]$.

The following lemma can help with the optimization of algorithms which use the incidence matrices. It implies that we do not have to consider the whole incidence matrices but only their parts.

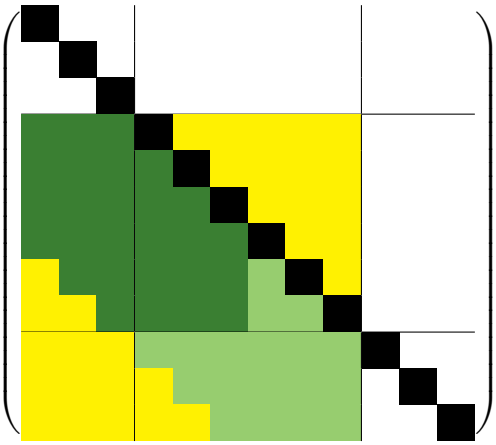


Figure 4.1: A schematic illustration of an incidence matrix M_Q in the proof of Lemma 6. The considered submatrix $\tilde{M}$ is colored as follows:

- diagonal elements of the incidence matrix;
- (i, j)'s in the submatrix $\tilde{M}$;
- remaining $(i + p, j + p)$'s in the submatrix $\tilde{M}$;
- "unimportant" elements in the submatrix $\tilde{M}$ (they cannot break the necessary and sufficient condition for the $(m, 2)$-solvability).

Lemma 6. Given a seed Q of span s and its incidence matrix M_Q, let us consider the $(m, 2)$-problem for some m. Let us denote $\ell = m - s$ and

$$\tilde{M} := M_Q \begin{bmatrix} \ell, \dots, m + \ell - 1 \\ 0, \dots, m - 1 \end{bmatrix} \in \{0, 1\}^{m \times m}.$$

Then the following statements are equivalent.

i) Q solves the $(m, 2)$-problem.

ii) For all indices

$$i \in \{0, 1, \dots, s - 1\},$$
$$j \in \{\max(-s + i, 0), \dots, \min(i + \ell - 1, m - 1)\},$$

there exists an index $p \in \{0, 1, \dots, \ell\}$ such that

$$\tilde{M}_{i+p,j+p} = 1 \qquad \text{or} \qquad \text{indices are out of range.}$$

Proof. The lemma is an easy consequence of Theorem 6 and Observation 2. A schematic illustration of an incidence matrix and its submatrix $\tilde{M}$ can be found in Figure 4.1. □

Remark 7. From now on, in illustrations of incidence matrices, we display only the "important elements", i.e., the elements, which are highlighted in Figure 4.1.

#	0	0	0					
-	1	1	0	1				
-	1	1	0	1	1			
-	1	1	0	1	1	1		
-	1	1	0	1	1	1	1	
			0	1	1	1	1	
			0	1	1	1	1	

Table 4.1: An individual beam in the Laser method.

Theorem 6 can be applied to construct seeds solving $(m, 2)$-problems. We fix m and s, then we try to use as many `#`'s as possible in order to maximize the weight of the seed while checking if the condition is still not broken.

Now let us introduce a visualization of this procedure, the so-called *Laser method*. A great advantage of this method is an easy usage for hand verification on a squared paper (or using the program presented in Section 6.1).

For the fixed m and s, we start with the seed $\texttt{-}^s$. Then, using some given algorithm, we add `#`'s one after one and check the incidence matrix, which can be illustrated as a square with a mirror formed by the diagonal elements. Each `#` from the seed Q corresponds to a "laser beam" of 0's. Such beam is targeted to the right, then it is reflected from the diagonal and changes the direction to the bottom as illustrated in Table 4.1.

Example 13 (Laser method). Consider $(14, 2)$-problem. A seed Q solves our problem if and only if its incidence matrix does not contain any block of the form

$$\begin{pmatrix} 0 & \star & \star & \star & \star & \star & \star \\ \star & 0 & \star & \star & \star & \star & \star \\ \star & \star & 0 & \star & \star & \star & \star \\ \star & \star & \star & 0 & \star & \star & \star \\ \star & \star & \star & \star & 0 & \star & \star \\ \star & \star & \star & \star & \star & 0 & \star \\ \star & \star & \star & \star & \star & \star & 0 \end{pmatrix}. \tag{4.1}$$

For instance, let us consider the seeds

$$Q^{(1)} = \texttt{###-\#--\#},$$
$$Q^{(2)} = \texttt{###-\#-\#\#}.$$

The first one solves the $(14, 2)$-problem (see Table 4.2), whereas the other one does not (see Table 4.3).

Remark 8. The incidence matrix of a seed can be rewritten using the matrix product of two vectors $\vec{x}$ and $(\vec{x})^\top$ as $M_Q = \vec{x} \cdot (\vec{x})^\top$, where $\vec{x} \in \{0, 1\}^{m+\ell}$ is a vector of the form of

$$\vec{x} = \begin{pmatrix} \vec{1} \\ \vec{y} \\ \vec{1} \end{pmatrix}$$

#	0	0	0	0	0	0	0							
#	0	0	0	0	0	0	0	0						
#	0	0	0	0	0	0	0	0	0					
-	1	1	1	1	1	1	0	0	0	1				
#	0	0	0	0	0	0	0	0	0	0	0			
-	1	1	1	1	1	1	0	0	0	1	0	1		
-	1	1	1	1	1	1	0	0	0	1	0	1	1	
#	0	0	0	0	0	0	0	0	0	0	0	0	0	0
							0	0	0	1	0	1	1	0
							0	0	0	1	0	1	1	0
							0	0	0	1	0	1	1	0
							0	0	0	1	0	1	1	0
							0	0	0	1	0	1	1	0
							0	0	0	1	0	1	1	0

Table 4.2: The Laser method for the seed $Q^{(1)}$ = `###-#--#` in the $(14, 2)$-problem. The condition (4.1) is satisfied, therefore, the seed solves the problem.

#	0	0	0	**0**	0	**0**	0							
#	0	0	0	0	**0**	0	**0**	0						
#	0	0	0	0	0	**0**	0	**0**	0					
-	1	1	1	1	1	1	**0**	0	**0**	1				
#	0	0	0	0	0	0	0	**0**	0	**0**	0			
-	1	1	1	1	1	1	0	0	**0**	1	**0**	1		
#	0	0	0	0	0	0	0	0	0	**0**	0	**0**	0	
#	0	0	0	0	0	0	0	0	0	0	**0**	0	**0**	0
							0	0	0	1	0	1	0	**0**
							0	0	0	1	0	1	0	0
							0	0	0	1	0	1	0	0
							0	0	0	1	0	1	0	0
							0	0	0	1	0	1	0	0
							0	0	0	1	0	1	0	0

Table 4.3: The Laser method for the seed $Q^{(2)}$ = `###-#-##` in the $(14, 2)$-problem. The condition (4.1) is not satisfied, therefore, the seed does not solve the problem. The elements breaking the condition are highlighted.

```
for s = 1, 2, ..., m − 1 do
    Q^(s) := -^s;
    create the m × m matrix for the Laser
    method;
    for i = 0, ..., s − 1 do
        if it is possible to add laser number
        i then
            add laser number i;
            (Q^(s))_i := #;
        end
    end
end
return
seed of the highest weight from {Q^(1), Q^(2), ..., Q^(m−1)}
```

Algorithm 2: Greedy algorithm on the Laser method for seeking $(m, 2)$-seeds.

with some vector $\vec{y} \in \{0,1\}^s$, where $\vec{1} \in \mathbb{N}^\ell$ is the all-ones vector of length ℓ.

Moreover, M_Q contains the prohibited block if and only if the discrete convolution of the $(\ell+1) \times (\ell+1)$ identity matrix $I_{\ell+1}$ and M_Q has some element equal to 0, i.e., we study for which vectors $\vec{x}$ of the prescribed form we can obtain

$$\min_{i,j} \left[I_{\ell+1} \star \vec{x}(\vec{x})^\top \right]_{i,j} \neq 0.$$

The correspondence between $\vec{x}$ (and $\vec{y}$ inside) and Q is following:

$$y_i = \begin{cases} 0 & \text{if } Q_i = \texttt{\#}, \\ 1 & \text{if } Q_i = \texttt{-}. \end{cases}$$

4.2 Greedy algorithm

The simplest algorithm to generate seeds solving the $(m, 2)$-problem using the Laser method can be based on the greedy algorithm. Its description can be found in Algorithm 2.

Let us derive the space and the time complexity of the algorithm. For every s we test s-times if it is possible to add a laser. Verification whether a laser can be added or its actual addition is $\mathcal{O}(m\ell)$. Hence the worst case time complexity of the algorithm is $\mathcal{O}(\sum_{s=1}^{m} sm(m-s)) = \mathcal{O}(m^4)$. The space complexity is determined by the size of matrix, which we use (by Lemma 6 we need only an $m \times m$ matrix), hence the space complexity is $\mathcal{O}(m^2)$.

Considered seeds in the case of the $(30, 2)$-problem can be found in Table 4.4. Best seeds for $(m, 2)$-problems with $m \in \{6, 7, \ldots, 40\}$ are presented in Table 4.5.

To demonstrate the power of this method, we extend [8, Table 2] to obtain Table 4.6. We see that for $m \leq 500$, it provides better results than the "asymptotic method" ([5], here presented in Remark 4) and the "perfect rulers method" from [8]. For small m we even obtain optimal seeds. It is caused by the fact that the greedy method constructs seeds as a repetitions of some good cyclic rulers (see Section 4.3).

s	ℓ	w	$Q^{(s)}$
2	28	2	`##`
3	27	3	`###`
4	26	4	`####`
5	25	5	`#####`
6	24	6	`######`
7	23	7	`#######`
8	22	8	`########`
9	21	9	`#########`
10	20	10	`##########`
11	19	10	`#########-#`
12	18	11	`#########-##`
13	17	12	`########-####`
14	16	11	`########-###--`
15	15	10	`#######-###----`
16	14	11	`#######-###----#`
17	13	12	`######-###----###`
18	12	13	`######-##----#####`
19	11	14	`#####-##---#####-##`
20	10	14	`#####-##---#####-##-`
21	9	13	`####-##---####-##---#`
22	8	14	`####-#---####-#---####`
23	7	14	`###-#--###-#--###-#--##`
24	6	15	`###-#--###-#--###-#--###`
25	5	13	`##-#--##-#--##-#--##-#--#`
26	4	11	`##---##---##---##---##---#`
27	3	9	`#--#--#--#--#--#--#--#--#--`
28	2	10	`#--#--#--#--#--#--#--#--#--#`
29	1	0	`----------`

Table 4.4: Seeds obtained by the greedy algorithm for the $(30, 2)$-problem. The best found seed is highlighted.

s: span of sought $Q^{(s)}$

ℓ: $m - s$

w: weight of $Q^{(s)}$

$Q^{(s)}$: seed obtained by the greedy algorithm

m	s	w	$Q^{(m)}$
6	2	2	`##`
7	2	2	`##`
8	2	2	`##`
9	3	3	`###`
10	3	3	`###`
11	5	4	`###-#`
12	4	4	`####`
13	8	5	`##-#--##`
14	6	5	`####-#`
15	9	6	`###-#--##`
16	10	7	`###-#--###`
17	10	7	`###-#--###`
18	12	8	`###-#--###-#`
19	12	8	`###-#--###-#`
20	12	8	`####-#---###`
21	13	9	`####-#---####`
22	16	10	`###-#--###-#--##`
23	17	11	`###-#--###-#--###`
24	17	11	`###-#--###-#--###`
25	19	12	`###-#--###-#--###-#`
26	16	12	`#####-##---#####`
27	16	12	`#####-##---#####`
28	18	13	`#####-##---#####-#`
29	19	14	`#####-##---#####-##`
30	24	15	`###-#--###-#--###-#--###`
31	24	15	`###-#--###-#--###-#--###`
32	26	16	`###-#--###-#--###-#--###-#`
33	24	16	`####-##---####-##---####`
34	22	16	`######-##----######-##`
35	25	17	`#####-##---#####-##---###`

Table 4.5: Results of the greedy algorithm for $(m, 2)$-problems. In the case of more seeds attaining the maximal weight, only the first seed is included.

s: span of $Q^{(m)}$

w: weight of $Q^{(m)}$

$Q^{(m)}$: the best seed obtained by the greedy algorithm for the $(m, 2)$-problem

m	16	32	48	64	80	96	200	300	400	500
AM	1	3	10	12	21	30	88	150	210	284
PR	7	14	23	31	41	50	109	166	224	283
GA	7	16	26	35	46	57	128	197	266	337
O	7	16	26	≥ 36						

Table 4.6: Comparison of methods for $(m, 2)$-design obtained as an extension of [8, Table 2] (weights of obtained seeds in dependence on m):

AM: "asymptotic method", the method generating asymptotically optimal seeds ([5], here described in Remark 4);

PR: "perfect rulers method" ([8]);

GA: greedy algorithm using the Laser method (Algorithm 2);

O: exhaustive search (optimal seeds).

Moreover, we can significantly improve the efficiency of the algorithm if we study outputs of the greedy algorithm in details (in Table 4.4 or using the enclosed software). We observe that, for given m and s, the found seeds are of the form of

$$\begin{gathered}\text{a prefix of } P^\omega \text{ of length } s, \quad \text{where } P = \texttt{\#}^a\texttt{-\#}^c\texttt{-}^d, \\ a = \left\lfloor \frac{\ell}{2} \right\rfloor, \quad c = \left\lfloor \frac{\ell - 1}{4} \right\rfloor, \quad \text{and} \quad d = \left\lfloor \frac{\ell}{4} \right\rfloor + 1.\end{gathered} \tag{4.2}$$

In the following proposition, we only prove that such seed solves the $(m, 2)$-problem. The proof is based on results from Section 4.3.

Proposition 6. Given an $(m, 2)$-problem and span $s \in \{1, 2, \ldots, m-1\}$, let us denote $\ell = m - s$. Then the seed (4.2) solves the $(m, 2)$-problem.

Proof. Let us denote the span of P by p. It follows from the formulae for a, c, and d that

$$p = \begin{cases} \ell & \text{if} \quad \ell \bmod p = 3, \\ \ell + 1 & \text{otherwise.} \end{cases}$$

Further, it is easy to verify that P is a cyclic ruler obtained by the greedy algorithm, as Lemma 3 describes. Since $p \leq \ell + 1$, Theorem 7 guarantees that any substring of P^ω of length s solves the $(m, 2)$-problem. □

Lemma 7. For $(m, 2)$-problems, the greedy algorithm does not provide asymptotically optimal seeds.

Proof. For contradiction, suppose that the greedy algorithm provides asymptotically optimal seeds, i.e., the limit (2.3) must hold.

We estimate the function

$$w(m, s) = \text{weight of the seed of span } s \text{ produced by the greedy algorithm}$$

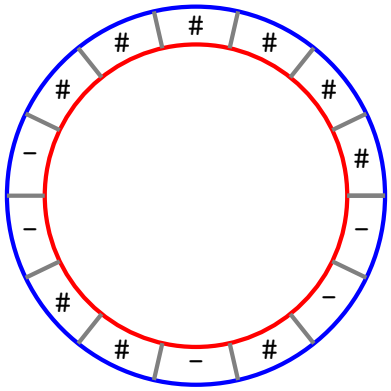


Figure 4.2: The cyclic ruler `######--##-#--`. With `-`'s as markers, any integer distance can be measured with such ruler.

by a function $\tilde{w}(m)$.

With respect to Proposition 6, we can make the following estimates.

$$w(m,s) \leq s \underbrace{\frac{\lfloor \frac{m-s}{2} \rfloor + \lfloor \frac{m-s-1}{4} \rfloor}{\lfloor \frac{m-s}{2} \rfloor + 1 + \lfloor \frac{m-s-1}{4} \rfloor + \lfloor \frac{m-s}{4} \rfloor + 1}}_{\text{density of \#}} + \underbrace{\left\lfloor \frac{m-s}{2} \right\rfloor + \left\lfloor \frac{m-s-1}{4} \right\rfloor}_{\text{correction of the ``tail''}}$$

$$\leq s \frac{\frac{m-s}{2} + \frac{m-s-1}{4}}{(\frac{m-s}{2} - \frac{1}{2}) + 1 + (\frac{m-s-1}{4} - \frac{3}{4}) + (\frac{m-s}{4} + \frac{1}{4})} + \frac{m-s}{2} + \frac{m-s-1}{4}$$

$$w(m,s) \leq \frac{3}{16} \frac{(4m-1)(m-s-\frac{1}{3})}{m-s-\frac{1}{4}} \leq \underbrace{\frac{3}{16}(4m-1)}_{=:\tilde{w}(m)}.$$

Since

$$\frac{w(m,s)}{m} \leq \frac{\tilde{w}(m,s)}{m} \leq \frac{3}{16}\left(4 - \frac{1}{m}\right)$$

holds for all admissible s, we have a contradiction with Corollary 1. Hence, in the case of $(m,2)$-problems, the greedy algorithm does not produce asymptotically optimal seeds. □

4.3 Cyclic rulers

This section is devoted to cyclic rulers — structures which can be used for designing $(m,2)$-seeds. We even have good reasons to believe that there exist no better seeds (i.e., seeds of higher weights) than those generated using cyclic rulers.

First we provide a definition and describe basic properties of cyclic rulers including a connection with perfect rulers. Then we describe some methods for cyclic ruler design. In the end, we generalize Lemma 4, which enables us to use cyclic rulers for $(m,2)$-seed design. We conclude with a conjecture on relation of cyclic rulers and all $(m,2)$-seeds.

4.3.1 Definition and basic properties

Definition 15. Let us have a seed P of span $p > 1$. We call it (complete) *cyclic ruler* if for all $d > 0$, there exists $i \geq 0$ such that

$$(P^\omega)_i = (P^\omega)_{i+d} = \texttt{-}.$$

Definition 16. A cyclic ruler P of span p is said to be *perfect*, if every other cyclic ruler of the same span has lesser or equal weight.

Example 14. We illustrate the cyclic ruler `######--##-#--`, which is even perfect, in Figure 4.2.

Definition 17. Let $P^{(1)}$ and $P^{(2)}$ by cyclic rulers. If there exists t such that

$$P^{(1)} = \sigma_t(P^{(2)}) \qquad \text{or} \qquad P^{(1)} = \text{ mirror image of } \sigma_t(P^{(2)}),$$

then we say that $P^{(1)}$ and $P^{(2)}$ are *dependent* cyclic rulers. Otherwise we call them *independent*.

As stated in the following lemma, cyclic rulers are exactly the seeds solving cyclic $(m, 2)$-problems.

Lemma 8. A seed P of span p is a cyclic ruler if and only if it solves the cyclic $(p, 2)$-problem.

Proof. $\Rightarrow$: Without loss of generality, it is sufficient to show that every similarity $L[0, d]$, where $d \in \{1, 2, \dots, p-1\}$, is cyclically detected by P.

Consider the mentioned similarity for some d. Then there exists $i \in \mathbb{N}$ such that $(P^\omega)_i = (P^\omega)_{i+d} =$ `-`. Therefore, $L[0, d]$ is cyclically detected by the seed $P_iP_{i+1}\cdots P_{i+p-1} = \sigma_{i \bmod p}(P)$. Thus $L[0, d]$ is cyclically detected by P.

$\Leftarrow$: Consider some positive d and denote $\tilde{d} = d \bmod p$. Hence $d = jp + \tilde{d}$ for some $j \geq 0$. By Definition 9, there must exist $t \in \{0, 1, \dots, p-1\}$ such that

$$(\sigma_t(P))_0 = (\sigma_t(P))_{\tilde{d}} = \texttt{-}.$$

Hence it holds

$$(P^\omega)_t = (P^\omega)_{t+\tilde{d}} = (P^\omega)_{t+d} = \texttt{-}.$$

□

Observation 3. Let P be a cyclic ruler. Then P`-` is a cyclic ruler. If $P_{p-1} =$ `#`, then $P_0P_1\cdots P_{p-2}$ is a cyclic ruler.

This observation implies that, by inserting `-` to a cyclic ruler at any position or by removing `#`, we obtain again a cyclic ruler.

The following lemma describes how to verify if a seed is a cyclic ruler with no need to perform any cyclic shift of it.

Lemma 9. A seed P of span p is a cyclic ruler if and only if the following statement holds:

$$(\forall d \in \{1, \dots, \lfloor \tfrac{p}{2} \rfloor\})\,(\exists i)\Big((P_i = P_{i+d} = \texttt{-}) \vee (P_i = P_{i+(p-d)} = \texttt{-})\Big). \tag{4.3}$$

Proof. The statement (4.3) is equivalent to the statement

$$(\forall d \in \{1, \dots, p\})\,(\exists i \in \{0, \dots, p-1\})((PP)_i = (PP)_{i+d} = \texttt{-}),$$

which is equivalent to the statement

$$(\forall d \in \{1, \dots, p\})\,(\exists i \in \{0, \dots, p-1\})((\sigma_i(P))_0 = (\sigma_i(P))_d = \texttt{-}).$$

The last statement is equivalent to the fact that P solves the $(p, 2)$-problem. □

As a simple corollary of the lemma we obtain what appears in [8, proof of Theorem 1].

Corollary 3. Let R be a complete ruler of span r. Then $P = R\#^{r-1}$ is a cyclic ruler.

4.3.2 Cyclic ruler design

Cyclic rulers can be constructed, for instance, using the following methods:

i) **Exhaustive search** (optimized, e.g., using the branch&bound method). It provides optimal cyclic rulers.

ii) **Asymptotic method** ([3], presented in Remark 3). It provides asymptotically optimal cyclic rulers but not optimal.

iii) **Greedy algorithm**. It produces cyclic rulers described in the following lemma. They are not asymptotically optimal (otherwise we would get a contradiction with Corollary 1 since $\lim_{m\to+\infty} \frac{w(m)}{m} \neq 1$).

Methods ii) and iii) are compared in Table 4.9.

Lemma 10. Let P be a cyclic ruler of span p produced by the greedy algorithm. Then it holds:

$$P = \texttt{\#}^a\texttt{-\#}^c\texttt{-}^d,$$

where

$$a = \left\lfloor \frac{p-1}{2} \right\rfloor, \quad c = \left\lfloor \frac{p-2}{4} \right\rfloor, \quad d = \left\lfloor \frac{p}{4} \right\rfloor + 1.$$

Sketch of the proof. We consider four possible cases in dependence on $p \bmod 4$ separately. For each case we can find an expression for a, b, and d. Then we can show that:

i) P is a cyclic ruler,

ii) $\texttt{\#}^{a+1}\texttt{-}^{p-a-1}$ is not a cyclic ruler (proof of this is same for all cases),

iii) every $\tilde{P}$ obtained from P by replacing any `-` by `#` is not a cyclic ruler. □

Remark 9. The problem of cyclic ruler design can be reformulated algebraically. For an arbitrary integer $p > 0$, consider the additive group $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$ and find its subset $X \subset \mathbb{Z}_p$ such that

$$X - X = \mathbb{Z}_p.$$

A cyclic ruler $P \in \{\texttt{\#}, \texttt{-}\}^p$ can be obtained from X in the following way:

$$P_i = \texttt{-} \quad \Leftrightarrow \quad i \in X$$

for all $i \in \{0, 1, \ldots, p-1\}$.

4.3.3 Seed design based on cyclic rulers

The following theorem can be used as a fundamental tool for construction of good seeds.

Theorem 7. If $\ell > 0$ and a seed P of span $p \in \{1, 2, \ldots, \ell+1\}$ solves the cyclic (p, k)-problem, then every substring Q of P^ω of span s solves the $(s+\ell, k)$-problem.

Proof. Denote $\delta = \ell + 1 - p$. By Lemma 4 the seed Q solves $(s + \ell - \delta, k)$-problem. Therefore, Q must solve also the $(s+\ell, k)$-problem. □

For every cyclic ruler, we define its characteristic function, which describes maximal weights of seeds generated by this cyclic ruler. This function if fully determined by the so-called characteristic vector.

Definition 18. For a cyclic ruler P of span p we define

i) the *characteristic function* $\chi_P : \mathbb{N} \to \mathbb{N}$ as:

$$\chi_P(i) = \max\{w(F) \mid F \text{ is a substring of } P^\omega \text{ of length } i\},$$

ii) the *characteristic vector* $\vec{V}(P) \in \mathbb{N}^p$ as:

$$\vec{V}(P) = \begin{pmatrix} \chi_P(0) \\ \chi_P(1) \\ \vdots \\ \chi_P(p-1) \end{pmatrix}.$$

Example 15. Consider the cyclic ruler $P =$ `#-####---` of span 10. Then the characteristic function has the following values:

i	$\chi_P(i)$	F	i	$\chi_P(i)$	F
0	0	ε	9	5	`#-####---`
1	1	`#`	10	6	`#-####---#`
2	2	`##`	11	7	`####---#-##`
3	3	`###`	12	8	`####---#-###`
4	4	`####`	13	9	`####---#-####`
5	4	`####-`	14	9	`####---#-####-`
6	5	`#-####`	15	10	`#-####---#-####`
7	5	`#-####-`	16	10	`#-####---#-####-`
8	5	`#-####--`	17	10	`#-####---#-####--`

Observation 4. Let P be a cyclic ruler of span p. Then the characteristic function can be obtained from the characteristic vector in the following way:

$$\chi_P(i) = w(P) \cdot \left\lfloor \frac{i}{p} \right\rfloor + \left(\vec{V}(P)\right)_{i \bmod p},$$

where $w(P)$ denotes the weight of P.

In order to distinguish cyclic rulers generating seeds of the same weights, it is useful to define an equivalence on the set of all cyclic rulers.

Definition 19. Two cyclic rulers $P^{(1)}$ and $P^{(2)}$ are said to be *equivalent* if they share the characteristic vector. We denote it $P^{(1)} \sim P^{(2)}$.

Observation 5. Two cyclic rulers share the same characteristic vector if and only if they share the same characteristic function.

Observation 6. Dependent cyclic rulers are equivalent.

Remark 10. There exist seeds which are equivalent but are not dependent, for instance:

$$P^{(1)} = \texttt{##----\#-\#-\#----},$$
$$P^{(2)} = \texttt{##--\#---\#-\#----}.$$

They both share the characteristic vector:

$$\vec{V}\left(P^{(1)}\right) = \vec{V}\left(P^{(2)}\right) = (0,1,2,2,2,3,3,3,3,4,4,5,5,5,5)^\top.$$

Now let us propose a method for seed design using cyclic rulers. Given an $(m, 2)$-problem and a seed span s, denote $\ell = m - s$. Let us consider a set of some precomputed cyclic rulers $\mathcal{P} = \{P^{(0)}, P^{(1)}, \ldots, P^{(r-1)}\}$ of spans lesser of equal to $\ell + 1$. Consider the set $\mathcal{Q}$ of all $(m, 2)$-seeds construable from the cyclic rulers from $\mathcal{P}$.

Let us show how to find a seed $Q \in \mathcal{Q}$ of span s of the highest possible weight. We choose $i \in \{0, 1, \ldots, r-1\}$ such that $\chi_{P^{(i)}}(s)$ is the maximal possible. We denote $P = P^{(i)}$ and $p = s(P^{(i)})$. Then we find the substring F of P of length $(i \bmod p)$ rich in #'s and take a substring Q of P^ω of length s such that F is a prefix of Q.

We believe that something like a converse implication of Theorem 7 holds, too. We present it as the following conjecture.

Conjecture 1. Let a seed Q of span s solve an $(m, 2)$-problem. Denote $\ell = m - s$. Then there exists a cyclic ruler P of span $p \leq \ell + 1$ such that the seed

$$\tilde{Q} = \text{ prefix of } P^\omega \text{ of length } s \tag{4.4}$$

satisfies $w(\tilde{Q}) \geq w(Q)$.

Remark 11. According to Theorem 7, every seed $\tilde{Q}$ of the form of (4.4) solves the $(m, 2)$-problem. Importance of Conjecture 1 lies in the fact that it would describe a structure of all seeds solving the $(m, 2)$-problem, including the optimal ones. For every such seed there would exist an associated cyclic ruler, from which we could create a seed of the smae span having the same or higher weight. In other words, every optimal seed would be obtainable by a permutation of letters of an optimal seed created from a cyclic ruler.

```
function
getHeaviestM2GreedySeed(m):
    // variables for the best seed
    s̃ := 0;
    w̃ := 0;
    õ := 0;
    ã := 0;
    c̃ := 0;
    d̃ := 0;
    for s = 1, 2, ..., m − 1 do
        a := ⌊ℓ/2⌋;
        c := ⌊(ℓ−1)/4⌋;
        d := ⌊ℓ/4⌋;
        p := a + 1 + c + d;
        o := ⌊s/p⌋;
        w := o · (a + c);
        s := s − o · p;
        if s ≤ a then
            w := w + s;
        else
            if s = a + 1 then
                w := w + a;
            else
                if s ≤ a + 1 + c then
                    w := w + s − 1;
                else
                    w := w + a + c;
                end
            end
        end
        // we found a better solution
        if w > w_max then
            s̃ := s;
            w̃ := w;
            õ := o;
            ã := a;
            c̃ := c;
            d̃ := d;
        end
    end
    return prefix of (#^ã -#^c̃ -^d̃)^(õ+1) of
    length s̃;
```

Algorithm 3: Greedy algorithm on the Laser method for seeking $(m, 2)$-seeds (improved version).

p	P	w	ρ
2	`--`	0	0.000
3	`#--`	1	0.500
4	`#---`	1	0.250
5	`##---`	2	0.400
	`#-#--`		
6	`##-#--`	3	0.500
7	`###-#--`	4	0.571
8	`###-#---`	4	0.500
	`###--#--`		
	`##-##---`		
	`##-#-#--`		
9	`####-#---`	5	0.556
	`####--#--`		
	`###-##---`		
	`###-#-#--`		
	`###-#--#-`		
	`##-##-#--`		
	`##-#-##--`		
10	`####-##---`	6	0.600
	`####-#-#--`		
	`###-##-#--`		
	`###-#-##--`		
11	`#####-##---`	7	0.636
	`#####-#-#--`		
	⋮		

Table 4.7: Independent optimal cyclic rulers.

p: span of the cyclic ruler

P: obtained cyclic ruler

w: weight of P

ρ: density of `#` in P, $\rho = \frac{w(P)}{s(P)}$

p	P	w	ρ
2	`--`	0	0.000
3	`#--`	1	0.333
4	`#---`	1	0.250
5	`##---`	2	0.400
6	`##-#--`	3	0.500
7	`###-#--`	4	0.571
8	`###-#---`	4	0.500
9	`####-#---`	5	0.556
10	`####-##---`	6	0.600
11	`#####-##---`	7	0.636
12	`#####-#-##--`	8	0.667
13	`######-#-##--`	9	0.692
14	`######-###----`	9	0.643
15	`#######-###----`	10	0.667
16	`#######-##-##---`	11	0.688
17	`########-##-##---`	12	0.706
18	`########-##-###--`	13	0.722
19	`#########-##-###--`	14	0.737
20	`#########-###-##----`	14	0.700

Table 4.8: Optimal cyclic rulers. Only one cyclic ruler is presented for each p.

p: span of sought cyclic ruler

P: obtained cyclic ruler

w: weight of P

ρ: density of `#` in P, $\rho = \frac{w(P)}{s(P)}$

m	8	16	32	64	128	200	300	400	500
GA	4	10	22	46	94	148	223	298	373
AM	3	9	21	49	106	172	266	361	456

Table 4.9: Comparison of approximation methods for cyclic ruler design — weight of obtained seeds in dependence on m and the used method.

GA: Greedy algorithm

AM: Asymptotically optimal method

CHAPTER 5

Generalization for more errors

We consider possible generalizations of the results from the previous chapter.

5.1 Laser method

The Laser method is easy to generalize. We should only remark that in more dimensions each # does not create a laser beam (a "kinked line") of zeros but a "kinked hyperplane of the dimension $k-1$" of zeros (and the mirror is also a hyperplane of the dimension $k-1$).

Definition 20 (Generalization of Definition 13)**.** For a string W of length n and a positive integer k, we define a map

$$\phi_k : \{0,1\}^n \to \{0,1\}^{\overbrace{n\times\cdots\times n}^{k \text{ times}}}$$

as

$$\left(\phi_k(W)\right)_{i_0,i_1,\ldots,i_{k-1}} = \begin{cases} 1 & \text{if } \bigwedge_{i\in\{i_0,i_1,\ldots,i_{k-1}\}} [W_i = \text{-}], \\ 0 & \text{otherwise,} \end{cases}$$

where the indices $i_0, i_1, \ldots, i_{k-1}$ are taken from $\{0,1,\ldots,n-1\}$.

Definition 21 (Generalization of Definition 14)**.** For a seed Q of span s and $k>1$, we define *k-dimensional incidence matrix* $\mathbb{M}_Q \in \{0,1\}^{\overbrace{s\times\cdots\times s}^{k \text{ times}}}$ as

$$\mathbb{M}_Q = \phi_k\left(\text{-}^\ell Q \text{-}^\ell\right).$$

Theorem 8 (Generalization of Theorem 6)**.** Consider a seed Q of span s, an (m,k)-problem and denote $\ell = m-s$. Then the following statements are equivalent:

i) the seed Q solves the (m,k)-problem;

ii) for all indices $i_0, i_1, \ldots, i_{k-1} \in \{0,1,\ldots,m-1\}$, there exists an index $p \in \{0,1,\ldots,\ell\}$ such that

$$\left(\mathbb{M}_Q\right)_{i_0+p,i_1+p,\ldots,i_{k-1}+p} = 1.$$

Proof. The proof is technically the same as for Theorem 6. □

5.2 Greedy algorithm

The greedy algorithm can be run with the multidimensional Laser method or directly with the definition of (m, k)-solvability as demonstrated in Algorithm 4.

```
for s = 1, 2, ..., m − 1 do
    Q^(s) := -^s;
    for i = 0, ..., s − 1 do
        (Q^(s))_i := #;
        if Q^(s) does not solve the
        (m, k)-problem then
            (Q^(s))_i := -;
        end
    end
end
return
seed with the highest weight from {Q^(1), Q^(2), ..., Q^(m−1)}
```

Algorithm 4: Greedy algorithm on the Laser method for seeking the optimal $(m, 2)$-seed.

The outputs of the greedy algorithm will probably have some periodical structure as they have in the case of two errors.

5.3 Cyclic rulers

Since cyclic rulers are the same as the seeds solving the cyclic $(m, 2)$-problems, the generalization of them are seeds solving (m, k)-problems for a higher k. It is also possible to use them for the seed design since Theorem 7 holds also for all $k > 1$. We can keep the definition of the characteristic function and the characteristic vector in order to use them for design of seeds from the given precomputed set of cyclic (m, k)-seeds.

Possible generalizations of Conjecture 1 have not been considered yet.

CHAPTER 6

Enclosed software

We present programs created for the purposes of this thesis, which can be found on the enclosed CD. The used languages are C++, Python, and HTML with JavaScript.

6.1 Laser method

Used languages: HTML with JavaScript
Directory: `1_Laser_method`

PROGRAMS

`Laser_GUI.html`
This program works in the environment of a web browser and is designed for the $(m, 2)$-problem analysis using the Laser method. Its screenshot can be found in Figure 6.1. User can set m and enter a specific seed, for which it generates the incidence matrix. Then it is possible to add/remove lasers (changing an individual `-` to `#` or vice versa) and increase or decrease m. Every occurrence of the prohibited diagonal string $\texttt{\#}^{\ell+1}$ (implying that the current seed does not solve the problem) is highlighted.

6.2 Seed design

Used languages: C++
Directory: `2_seed_design`

PROGRAMS

1) **`main_branchbound.cpp`**
It seeks optimal $(m, 2)$-seeds using the branch&bound method for the exhaustive search.

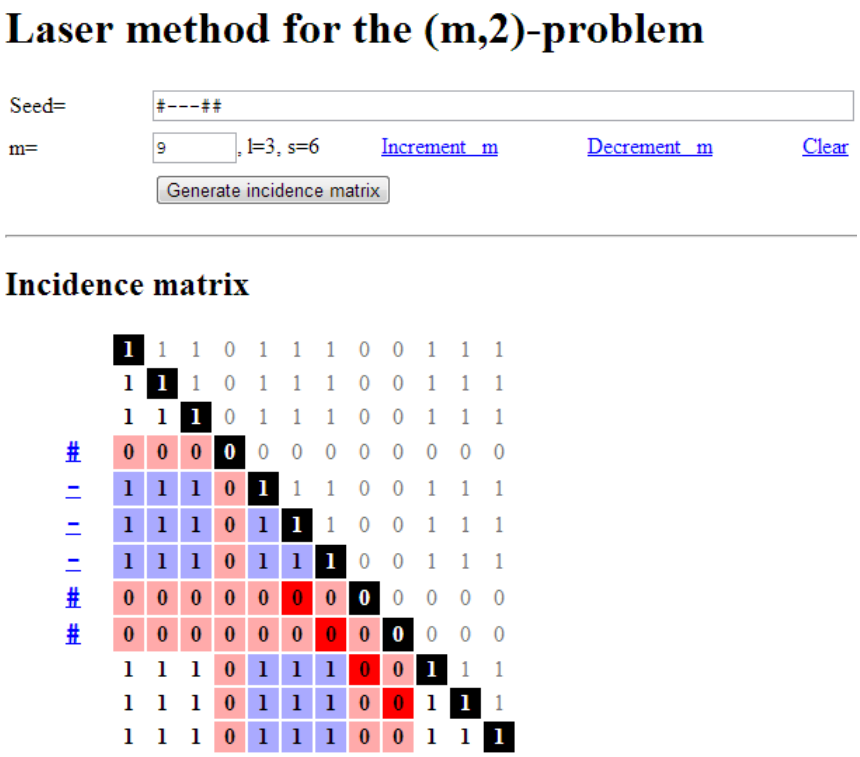


Figure 6.1: GUI for the Laser method

2) `main_greedy_laser.cpp`
It seeks $(m, 2)$-seeds using the greedy algorithm with the Laser method (see Algorithm 2).

3) `main_greedy_imp.cpp`
It seeks the same $(m, 2)$-seeds as the greedy algorithm does but in more effective way (using (4.2)).

CLASSES AND FUNCTIONS

1) `seeds.cpp`, `seeds.h`
Auxiliary functions for seeds.

   i) `int seedWeight(const string &seed);`
   It returns the weight of the given seed.

   ii) `void nextPermutation(string &similarity);`
   It changes the given (m, k)-similarity to the next one (e.g., 111010000 $\rightarrow$ 111001000).

   iii) `bool seedSolvesMkroblem(const string &seed, int m,int k);`
   Test whether the given seed solves the given (m, k)-problem.

2) `LaserMethod.cpp`, `LaserMethod.h`
Class `LaserMethod`.

   i) `LaserMethod(int m, int s);`
   Constructor. It initialize the incidence matrix for the $(m, 2)$-problem and seeds of span s.

ii) `LaserMethod(const LaserMethod& anotherLaserMethod);`
Copy constructor.

iii) `~LaserMethod();`
Destructor, it frees all allocated memory.

iv) `int getM();`
It returns m.

v) `void addLaser(int laser);`
It adds the specified laser.

vi) `void removeLaser(int laser);`
It removes the specified laser.

vii) `bool isLaserAddable(int laser);`
It tests whether the given laser is addable (i.e., if the seed associated to the stored incidence matrix will solve the $(m, 2)$-problem also when containing `#` at position *laser*).

viii) `void print(bool debug);`
It prints the stored incidence matrix in the debug or in the non-debug mode.

ix) `void print();`
It prints the stored incidence matrix in the non-debug mode.

x) `int weight();`
It returns the weight of the seed associated to the stored incidence matrix.

xi) `operator string();`
It returns the seed associated to the stored incidence matrix.

- **`LM_BranchBound.cpp`**, **`LM_BranchBound.h`**
Class `LM_BranchBound`.

i) `LA_BranchBound(int m);`
Constructor. It initializes the incidence matrix for the $(m, 2)$-problem and seeds of span s.

ii) `~LA_BranchBound();`
Destructor. It frees all used memory.

iii) `void performBB(int s);`
It performs the branch&bound search for the given span s and continuously prints the current best seed. The branch&bound method perform the depth-first search in the graph in Figure 6.2. A branch is cut if:

i) the parental node does not solve the $(m, 2)$-problem;

ii) the currently found best seed has greater or equal weight than the seeds in the branch could have.

6.3 Cyclic ruler design

Used languages: Python 3
Directory: `3_cyclic_ruler_design`

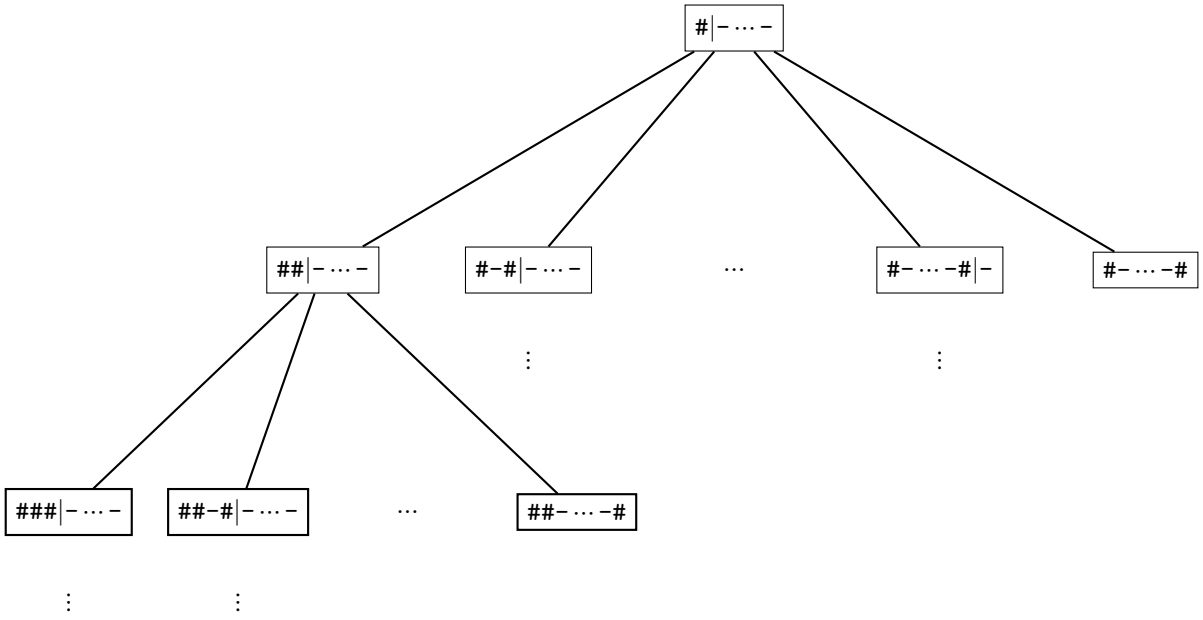


Figure 6.2: A tree of seeds for the branch and bound method.

PROGRAMS

1) `CycRul_as_opt.py`
   It finds cyclic rulers using the method described in Remark 3.
2) `CycRul_brute_force.py`
   It seeks cyclic rulers using the brute force algorithm (it probes all strings over $\{\texttt{\#}, \texttt{-}\}$ of a given length).
3) `CycRul_greedy.py`
   It seeks cyclic rulers using the greedy algorithm.
4) `CycRul_greedy_impr.py`
   It seeks the same cyclic rulers as the greedy algorithm does (the method given by Lemma 10 is used).

CLASSES AND FUNCTIONS

`CycRul___lib___.py`
Auxiliary functions.

i) `is_cycl_ruler(P)`
   It tests whether the given seed P is a cyclic ruler.
ii) `cycl_rulers_dep(CR1,CR2)`
   It tests whether the given circular rulers $CR1$ and $CR2$ are dependent, i.e., $CR2$ can be obtained from $CR1$ by a cyclic shift or as a mirror image of a cyclic shift.
iii) `get_independent(CRs)`
   For the given set of cyclic rulers CRs, it returns the biggest possible subset of CRs containing only the independent cyclic rulers.

iv) `tex_line(CR)`
It prints information about the given cyclic ruler in LaTeX array format (namely the cyclic ruler, its span and weight, and the density of `#` in the cyclic ruler.

CHAPTER 7

Conclusion

7.1 Results

This thesis deals with lossless seeds for approximate string matching. First we introduced basic definitions and made a survey of existing results for the case of a single seed and one hit. Then we systematically studied seed design for problems with one and two errors.

The most important original results presented in this thesis are:

i) Theorem 5 as a new necessary and sufficient criterion for the solvability of $(m, 1)$-problems, which can be used together with greedy algorithm for construction of optimal $(m, 1)$-seeds (see Chapter 3).

ii) The Laser method as a new necessary and sufficient criterion for the solvability of $(m, 2)$-problems (see Section 4.1).

iii) Analysis of the greedy algorithm in $(m, 2)$-problems (see Section 4.2).

iv) Approach of $(m, 2)$-seed design based on cyclic rulers (see Section 4.3) including:

- Theorem 7, which is a generalization of the results from the paper [3],
- Conjecture 1 implying (if it holds) that for every m, an optimal $(m, 2)$-seed can be obtained from a cyclic ruler.

v) Computer programs located on the enclosed CD (see Chapter 6).

7.2 Further work

During the work on this thesis, a few issues, which have not been solved sufficiently and require further study, appeared. Let us mention the most important of them.

i) To prove Conjecture 1 or find a counterexample.

   Remark that in the case when the conjecture did not hold, we still could construct $(m, 2)$-seeds from cyclic rulers. Nevertheless, we would have no guarantee that we could obtain optimal ones.

ii) To describe properties of cyclic rulers — to answer, for instance, the following questions.

- Is the perfect cyclic ruler design an NP-complete task (as it holds for the perfect ruler design)?
- Is there any generating function for some perfect cyclic rulers (as an analogy to the function given by (2.4) for the Wichmann rulers)?
- For which characteristic vectors do the corresponding cyclic rulers exist and how can we find them?

Bibliography

[1] S. Burkhardt, J. Kärkkäinen. Better filtering with gapped q-grams. *Fundamenta Informaticae* **56**(1), pp. 51–70, 2003.

[2] B. Ma, J. Tromp, and M. Li. PatternHunter. Faster and more sensitive homology search. *Bioinformatics* **18**(3), pp. 440–445, 2002.

[3] G. Kucherov, L. Noé, and M. Roytberg. Multi-seed lossless filtration. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **2**(1), pp. 51–61, 2005.

[4] L. Noé and G. Kucherov. YASS: enhancing the sensitivity of DNA similarity search. *Nucleic Acid Research* **33**, pp. 540–543, 2005.

[5] M. Farach-Colton, G. M. Landau, S. C. Sahinalp, and D. Tsur. Optimal spaced seeds for faster approximate string matching. *Journal of Computer and System Sciences* **73**(7), pp. 1035–1044, 2007.

[6] F. Nicolas and É. Rivals. Hardness of optimal spaced seed design. *Journal of Computer and System Sciences* **74**(5), pp. 831–849, 2008.

[7] H. Lin, Z. Zhang, M. Q. Zhang, B. Ma, and M. Li. Zoom! zillions of oligos mapped. *Bioinformatics* **24**(21), pp. 2431–2437, 2008.

[8] L. Egidi, G. Manzini. Spaced Seeds Design Using Perfect Rulers. In R. Grossi, F. Sebastiani, and F. Silvestri, editors. *String Processing and Information Retrieval, 18th International Symposium, SPIRE 2011, Pisa, Italy, October 17-21, 2011. Proceedings,* volume 7024 of *Lecture Notes in Computer Science*, pp. 32–43, Springer, 2011.

[9] L. Egidi, G. Manzini. Better spaced seeds using quadratic residues. *Journal of Computer and System Sciences* **79**(7), pp. 1144–1155, 2013.

[10] L. Noé. Spaced seeds bibliography. `http://www.lifl.fr/~noe/spaced_seeds.html` [April 21, 2013].

[11] B. Wichmann. A note on restricted difference bases. *Journal of the London Mathematical Society* **1**(1), pp. 465–466, 1963.

Index