APPLICATION OF SEMI-LOCAL LCS TO STRING APPROXIMATE MATCHING*

DIANNE DOE[†], PAUL T. FRANK[‡], AND JANE E. SMITH[‡]

Abstract. We present an application of semi-local lcs to approximate string matching by developing a new algorithm and improving the existing one. Our result is based on the utilization of the underlying algebraic structure of semi-local lcs with the usage of the novel data structure for submatrix maximum queries in Monge matrices. This gives two algorithms with the following running time and space complexity. TODO. The improvement of the existing algorithm not only preserves all properties but also outperforms in practice.

In addition, we show that the algorithm for semi-local lcs based on sticky braid multiplication is not perform well with the current complex recursive structure.

Key words. semi-local lcs, monge matrix, range queries, approximate matching, near-duplicate detection

AMS subject classifications. 68Q25, 68R10, 68U05

1. Introduction.

Approximate string matching is an important task in many fields such as computational biology, signal processing, text retrieval and etc. It also refers to a duplicate detection subtask.

In general form it formulates as follows: Given some pattern p and text t need to find all occurrences of pattern p in text t with some degree of similarity.

There are many algorithms that solve the above problem. Nonetheless, the number of algorithms sharply decreases when the algorithm needs to meet some specific requirements imposed by running time, space complexity or specific criterion for the algorithm itself. For example, recently there was developed an approach for interactive duplicate detection for software documentation[]. The core of this approach is an algorithm that detects approximate clones of a given user pattern with a specified degree of similarity. The main advantage of the algorithm is that it meets a specific requirement of completeness. Nonetheless, it has an unpleasant time complexity.

The algorithm for approximate detection utilizes mainly algorithm for solving the longest commons subsequence (LCS) problem. The longest common subsequence is a well-known fundamental problem in computer science that also has many applications of its own. The major drawback of it that it shows only the global similarity for given input strings. For many tasks, it's simply not enough. The approximate matching is an example of it.

There exist generalazation for LCS called *semi-local LCS* [] which overcome this constraint. The effective theoretical solutions for this generalized problem found applications to various algorithmic problems such as bla bla add cited. For example, there has been developed algorithm for approximate matching in the grammar-compresed strings[].

Although the algorithms for *semi-local LCS* have good theoretical properties, there is unclear how they would behave in practice for a specific task and domain.

To show the applicability of semi-local lcs on practice we developed several algorithms based mainly on it and the underlying algebraic structure. As well as devel-

*Submitted to the editors DATE.

Funding: This work was funded by the Fog Research Institute under contract no. FRI-454.

[†]Imagination Corp., Chicago, IL (ddoe@imag.com, http://www.imag.com/~ddoe/).

[‡]Department of Applied Mathematics, Fictional University, Boise, ID (ptfrank@fictional.edu, jesmith@fictional.edu).

oping new algorithms we improve and significantly outperform the existing one for interactive duplicate detection for software documentation []. It should be noted that improvement preserves all properties of this algorithm. Do we need to state that ant algo is slow for current strucute of algorithm

The paper is organized as follows. Blablabla **??**, our new algorithm is in **??**, experimental results are in **??**, and the conclusions follow in **??**.

2. Preliminaries.

2.1. Approximate matching.

Describe approximate matching formally

2.2. Semi-local lcs.

Describe semi-local lcs (definition), algorithms that solves (steady and and braid reducing)

2.3. Monge matrix.

Describe monge property

Say about range queries (about soda12, soda14 and new result that we will be used)

2.4. Near-duplicate detection algorithm.

Describe luciv algo

3. Related work.

?????

could mention about approximation. Need discuss

4. Algorithm for near duplicate detection.

Describe our algorithm and shows our optimization.

Proof all properties satisified.

Make note that not contraint on other metric???

5. CutMax a new approximate mathing algorithm.

Descirbe algroithm.

Present algo implementation with sparse table. Say that bad.

Describe optimization via monge property.

Describe complexity

6. Evaluation.

Semi-local algorithms. Show perfomance between lcs and semi-local lcs??? and poor perfomance of recursive algorithm based on steady ant?

Approximate matching algorithms. Show outperforming for different cases between luciv and our algorithm.

Show quality betwee our new algo and luciv algo (our should be better)

Show that sparse table bad when large?

7. Conclusion.

Say may be succesfully be applied on practice (showed by algorithm luciv updated)

Open problem.− >

Say that need to implement with monge2020 (what we not finished)

Improve algo based on recursive steady ant. Because it's critical for algos based on it.

df[1]

Acknowledgments. We would like to acknowledge the assistance of volunteers in putting together this example manuscript and supplement.

REFERENCES

[1] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 4th ed., 2013.