# The Set-Set Closest Common Subsequence Problem ${ }^{1}$ 

Gabriela Andrejková<br>Department of Computer Science, Faculty of Science, P. J. Šafárik University, Jesenná 5, 04154 Košice, Slovakia<br>e-mail: andrejk@kosice.upjs.sk


#### Abstract

Efficient algorithm is presented that solves a general case of the Common Subsequence Problem, in which both input strings contain sets of symbols with membership values in the sets. The problem arises from a searching of the sets of most similar strings.


Key words: Subsequence, common subsequence, measure of the string, dynamic programming, design and analysis of algorithms.

## 1 Introduction

The motivation to the Closest Common Subsequence (CCS Problems) can be found in the typing of a text on the keyboard. The following mistakes can be made in typing some string: (1) Typing a different character, usually from the neighbour area of the given character. (2) Inserting a single character into the source string. (3) Omitting (skipping) any single source character. (4) Transposition of two elements. It means, we have some words with mistakes. The problem is how to find the strings they are very similar very closed to the exact strings. Very important role has here the common subsequence of similar strings.

The common subsequence problem of two strings is to determine one of the subsequences that can be obtained by deleting zero or more symbols from each of the given strings. It is possible to demand some additional properties for the common subsequence. Usually, it is the greatest length of the common subsequence, but we can consider some different measures for the common subsequence.

The longest common subsequence problem (LCS Problem) of two strings is to determine the common subsequence with the maximal length. Algorithms for this problem can be used in chemical and genetic applications and in many problems concerning data and text processing [4, 8, 10]. Further applications include the string-to-string correction problem [8] and determining the measure of differences between text files [4]. The length of the longest common subsequence (LLCS Problem) can determine

[^0]the measure of differences (or similarities) of text files. The simulation method for the approximate strings and sequence matching using the Levenstein metric can be found in J. Holub [7] and the algorithm for the searching of the subsequences is in Z . Troníček and B. Melichar [11].
D. S. Hirschberg and L. L. Larmore [6] have discussed a generalization of LCS Problem, which is called Set-Set LCS Problem (SSLCS Problem). In this case both strings are strings of subsets over an alphabet $\Omega$. In the paper [6] is presented the $O(m \cdot n)$ time algorithm for the general SSLCS Problem.
In this paper we present algorithms for more general case of the Common Subsequence Problem, it means Closest Common Subsequence Problem SSCCS Problem for two strings of symbol sets with membership values of elements in the sets.

## 2 Basic Definitions

In this section, some basic definitions and results concerning to CCS Problem, SCCS and SSCCS Problem are presented.
Let $\Omega$ be a finite alphabet, $|\Omega|=s, P(\Omega)$ the set of all subsets of $\Omega,|P(\Omega)|=2^{s}$.
Let $A=a_{1} a_{2} \ldots a_{m}, a_{i} \in \Omega, 1 \leq i \leq m$ be a string over an alphabet $\Omega$, where $|A|=m$ is the length of the string A.
The string $C \in \Omega^{*}, C=c_{1} \ldots c_{p}$ is a subsequence of the string $A=a_{1} \ldots a_{m}$, if a monotonous increasing sequence of natural numbers $i_{1}<\cdots<i_{p}$ exists such that $c_{j}=a_{i_{j}}, 1 \leq j \leq p$. The string $C$ is a common subsequence of two strings $A, B$ if $C$ is a subsequence of $A$ and $C$ is a subsequence of $B .|C|$ is the length of the common subsequence. The classical problem to find the longest common subsequence is defined and solved in Hirschberg [5]. In the classical problem, each element in the string is in his position as full member, but sometimes we are not sure about it in texts. The element should be in his position with $70 \%$, it means, the element is in his position with 0.7 membership value. Sometimes, we can suppose that in some position should be one element of some set of elements with membership values.
Let $\mu_{A}\left(a_{i}\right) \in(0,1\rangle, 1 \leq i \leq m$, be some membership values of elements in the string A. The pair $\left(A, \mu_{A}\right)$ is the string $A$ with the membership function $\mu_{A}, m$-string $\mu A$ for short. $\operatorname{Val}(\mu A)$ is a measure of $\mu A$ defined by the (1).

$$
\begin{equation*}
\operatorname{Val}(\mu A)=\sum_{i=1}^{m} \mu_{A}\left(a_{i}\right) \tag{1}
\end{equation*}
$$

The string $\mu C=\left(C, \mu_{C}\right)$ is a subsequence with the membership function $\mu_{C}$, shortly $m$-subsequence of the m-string $\mu A$ if $C$ is a subsequence of the string $A$ and $0<$ $\mu_{C}\left(c_{t}\right) \leq \mu_{A}\left(a_{i_{t}}\right)$, for $1 \leq t \leq p$. The m-subsequence $\mu C$ is a closest $m$-subsequence if $\operatorname{Val}(\mu C)=\Sigma_{j=1}^{p} \mu_{C}\left(c_{j}\right)=\Sigma_{j=1}^{p} \mu_{A}\left(a_{i_{j}}\right)$.
The string $\mu C$ is a common m-subsequence of two m-strings $\mu A$ and $\mu B$ if $\mu C$ is a m-subsequence of $\mu A$ and $\mu C$ is a m-subsequence of $\mu B$.
The string $\mu C$ is a closest common m-subsequence of the m-strings $\mu A$ and $\mu B$ if $\mu C$ is a common m-subsequence with the maximal value $\operatorname{Val}(\mu C)$. It means, if $\mu D$ is a common m-subsequence of the strings $\mu A$ and $\mu B$ then $\operatorname{Val}(\mu D) \leq \operatorname{Val}(\mu C)$.

If $\mu C$ is a closest common m-subsequence of the m-strings, $\mu A$ and $\mu B$ then $\mu_{C}\left(c_{t}\right)=$ $\min \left\{\mu_{A}\left(a_{k_{t}}\right), \mu_{B}\left(b_{l_{t}}\right)\right\}$, for $1 \leq t \leq p$.

The CCS Problem: Let $\mu A$ and $\mu B$ be m-strings. To find a closest common subsequence of the m-strings $\mu A$ and $\mu B, C C S(\mu A, \mu B)$ for short.
The MCCS Problem is to find the measure of CCS m-string, MCCS for short. It means, $M C C S(\mu A, \mu B)=\operatorname{Val}(C C S(\mu A, \mu B))$.
Algorithms for CCS and MCCS Problem Andrejková [2].


Figure 1. The closest common subsequence of two m-strings $A$ and $B$.

Example 1. $\Omega=\{a, b, c\}, A=a b a a b a c a b, m=9, \mu_{A}=(.9, .9, .6, .5, .2, .8, .4, .6, .5)$, $B=a b c d b c b, n=7, \mu_{B}=(.6, .6, .3, .4, .9, .5, .6)$. The string $C=a b c b$ is a subsequence, $C^{\prime}=a b b c b$ is the longest common subsequence of the strings $A$ and $B$, and $\mu C^{\prime \prime}$, $C^{\prime \prime}=a b c b, \mu_{C^{\prime \prime}}=(.6, .9, .4, .5)$ is the closest common subsequence of the m-strings $\mu A$ and $\mu B, \operatorname{Val}(\mu C ")=\operatorname{MCCS}(\mu A, \mu B)=2.4$ as it is shown in Figure 1.
$A$ string of sets $\mathcal{B}$ over an alphabet $\Omega$, set-string for short, is any finite sequence of sets from $P(\Omega)$. Formally, $\mathcal{B}=B_{1} B_{2} \ldots B_{n}, B_{i} \in P(\Omega), 1 \leq i \leq n, n$ is the number of sets in $\mathcal{B}$. The length of the symbol string described by $\mathcal{B}$ is $N=\sum_{i=1}^{n}\left|B_{i}\right|$. A string of symbols $C=c_{1} c_{2} \ldots c_{p}, c_{i} \in \Omega, 1 \leq i \leq p$, is a subsequence of symbols (subsequence, for short) of the set-string $\mathcal{B}$ if there is a nondecreasing mapping $F:\{1,2, \ldots, p\} \rightarrow$ $\{1,2, \ldots, n\}$, such that

1. if $F(i)=k$ then $c_{i} \in B_{k}$, for $i=1,2, \ldots, p$
2. if $F(i)=k$ and $F(j)=k$ and $i \neq j$ then $c_{i} \neq c_{j}$.

Let $\mathcal{A}=A_{1} \ldots A_{m}, \mathcal{B}=B_{1} \ldots B_{n}, 1 \leq m \leq n$, be two set-strings of sets over an alphabet $\Omega$. The string of symbols $C$ is a common subsequence of symbols of $\mathcal{A}$ and $\mathcal{B}$ is $C$ a subsequence of symbols of $\mathcal{A}$ and $C$ is a subsequence of symbols of the set-string $\mathcal{B}$.
As similar as for strings, let define $m$-set as a set with membership function defined on its elements.
Let $\mu_{B_{i}}, i=1,2, \ldots, n$ be the membership functions of the sets $B_{i}, i=1,2, \ldots, n$ in the string $\mathcal{B}$. It means, $\mu \mathcal{B}=\mu B_{1} \mu B_{2} \ldots \mu B_{n} . \mu \mathcal{B}$ is the m-set-string $\mathcal{B}$ of m-sets $B_{i}, i=1,2, \ldots, n$ with the membership functions $\mu_{B_{i}}$, m-set-string $\mu \mathcal{B}$ for short. The weight of the $m$-set $B \in \mathcal{P}(\Omega)$ with membership function $\mu_{B}$ is defined by

$$
\begin{equation*}
W(B)=\sum_{x \in B} \mu_{B}(x) \tag{2}
\end{equation*}
$$

A string $\mu C$ is a m-subsequence of the m-set-string $\mu \mathcal{B}$ if (1) $\mu C$ is the subsequence of symbols of the set-string $\mathcal{B}$ and (2) if $c=c_{i}, c_{i} \in B_{k}$ then $\mu_{C}\left(c_{i}\right) \leq \mu_{B_{k}}(c)$.
The m-string $\mu C$ is a common m-subsequence of the m-set-strings $\mu \mathcal{A}$ and $\mu \mathcal{B}$ if $\mu C$ is m-subsequence of $\mu \mathcal{A}$ and $\mu C$ is m-subsequence of $\mu \mathcal{B}$.

The string $\mu C$ is $a$ closest common m-subsequence of the m-set-strings $\mu \mathcal{A}$ and $\mu \mathcal{B}$ if $\mu C$ is a common subsequence with maximal value $\operatorname{Val}(\mu C)$. Note that $\mu C$ is not in general unique.

The SSCCS Problem: Let $\mu \mathcal{A}, \mu \mathcal{B}$ be two m-set-strings. The Set-Set Closest Common Subsequence problem of the m-set-strings $\mu \mathcal{A}$ and $\mu \mathcal{B},(\operatorname{SSCCS}(\mu \mathcal{A}, \mu \mathcal{B})$ for short, consists of finding a closest common m-subsequence $\mu C$ with the maximal value $\operatorname{Val}(\mu C)$.
The MSSCCS Problem consists of finding the measure of $S S C C S$ m-setstring, $\operatorname{MSSCCS}(\mu \mathcal{A}, \mu \mathcal{B})$ for short.
It means, $\operatorname{MSSCCS}(\mu \mathcal{A}, \mu \mathcal{B})=\operatorname{Val}(\operatorname{SSCCS}(\mu \mathcal{A}, \mu \mathcal{B}))$, $\diamond$

Figure 2. The closest common m-subsequence of two m-set-strings $\mathcal{A}$ and $\mathcal{B}$.

Example 2. Let $\mathcal{A}=\{a, d\}\{c, a, d\}\{e, b, a\}, m=3, \mu_{A_{1}}=(.7, .3), \mu_{A_{2}}=(.6, .4$, $.5), \mu_{A_{3}}=(.6, .3, .8), \mathcal{B}=\{d, e, c\}\{a, d, e\}\{b, d, c\}\{b, d\}, n=4 . \mu_{B_{1}}=(.4, .3, .5), \mu_{B_{2}}=$ $(.7, .6, .8), \mu_{B_{3}}=(.9, .5, .7), \mu_{B_{4}}=(.5, .3)$. The membership values are described according to the named order in the set. For example, $\mu_{A_{1}}(a)=0.7, \mu_{A_{1}}(d)=0.3$. Then $\operatorname{MSSCCS}(\mu \mathcal{A}, \mu \mathcal{B})=2.4$ as it is shown in the Figure 2.

## 3 ALGORITHM FOR MSSCCS Problem

The basic idea of the algorithm starts from the definition of MSSCCS Problem.

$$
\begin{align*}
& \operatorname{MSSCCS}(\mu \mathcal{A}, \mu \mathcal{B})=\max _{\mu C}\{\operatorname{Val}(\mu C): \mu C \text { is the common m-subsequence } \\
& \text { of } \mu \mathcal{A} \text { and } \mu \mathcal{B}\} \tag{3}
\end{align*}
$$

In the following part of the paper we will use the m-set only and for a simpler description we will omit the symbol $\mu$ in the names of sets.
A flattening of a sequence of sets is defined as a concatenation, in order of the sequence, of strings formed by some permutation of individual elements of the sets in the sequence. For example, a flattening of the m-set-string $\mathcal{A}$ in Example 2 is
$\mathcal{A}_{f l 1}=$ dabaceba, $\mu_{\mathcal{A}_{f l 1}}=(.3, .7, .5, .4, .6, .6, .3, .8)$ and so is $\mathcal{A}_{f l 2}=d a a c b b a e, \mu_{\mathcal{A}_{f l 2}}=$ $(.3, .7, .4, .6, .5, .3, .8, .6)$.
If we have some flattening of both set-strings then it is possible to apply the MCCS algorithm, Andrejková [2]. It is necessary to compute $M C C S$ values of all pairs of all flattenings both set-strings but it is too much time consuming.
If we have the flattening of one set-string and the second is a set-string then it is possible to use the MSCCS algorithms. But it is necessary to compute MSCCS value for all flattenig of one string. It is to much time consuming too. Both algorithms have an exponential time complexity.
It is possible to use the following algorithm of polynomial time complexity. The algorithm works in two steps:

1. to create the string of symbols for each of set-string; each set can be encoded as the string of all permutations of its elements (the length of such string is $k^{2}-2 \cdot k+4, \mathrm{k}$ is the number of elements in set [9]); for example, the shortest m -string of elements in the m -set-string $\mathcal{A}$ in example 2 is dadcabcabcbeabeab and so is adacabcabcebaebae.
2. to apply the MCCRS algorithm, Andrejková [1] for the two in the previous step constructed m-strings (each element of the m-set can be used once at most);

The algorithm works in polynomial time: $O\left(M^{2} \cdot N^{2} \cdot K\right)$, where $M=\sum_{i=1}^{m}\left|A^{i}\right|$, $N=\sum_{j=1}^{n}\left|B^{j}\right|$, and $K$ is the number of elements in the closest common restricted subsequence.

We formulate the following algorithm with the better time complexity according to Hirshberg's idea for SSLCS Problem [6]. The algorithm works with intersection, union and equivalence, difference of m-sets. It is possible to use many definitions of them but the following $[3,12]$ are more obvious:
Let $A, B$ be the m-sets with membership functions $\mu_{A}, \mu_{B}$, and $x \varepsilon A$ explains a membership of the element $x$ to the m-set $A, \mu_{A}(x)>0$, then

1. intersection " $\bigcap_{m}$ " of two m-sets is defined:

$$
A \cap_{m} B=_{d e f}\{x \mid x \varepsilon A \wedge x \varepsilon B\}, \mu_{A \cap_{m} B}(x)=\min \left\{\mu_{A}(x), \mu_{B}(x)\right\}
$$

2. union " $\cup_{m}$ " of two m-sets is defined:

$$
A \cup_{m} B=_{\text {def }}\{x \mid x \varepsilon A \vee x \varepsilon B\}, \mu_{A \cup_{m} B}(x)=\max \left\{\mu_{A}(x), \mu_{B}(x)\right\}
$$

3. equivalence $"={ }_{m}$ " of two m -sets is defined:

$$
A={ }_{m} B \Leftrightarrow x \varepsilon A \wedge x \varepsilon B \wedge x \varepsilon A \cap_{m} B \wedge x \varepsilon A \cup_{m} B
$$

4. difference " $-m$ " of two $m$-sets is defined:

$$
A-_{m} B=_{\text {def }}\{x \mid x \varepsilon A \wedge x \notin B\}, \mu_{A-B}(x)=\mu_{A}(x)
$$

5. $A$ is m-subset of $B, A \subseteq_{m} B, \quad$ if and only if $\forall x \varepsilon A$ is fulfilled $x \varepsilon B$ and $0<$ $\mu_{A}(x) \leq \mu_{B}(x)$.

### 3.1 Description of the simple algorithm A.

For convenience, we define $A_{0}=B_{0}=\emptyset$.
We define $\operatorname{Ent}(i, j)$ to be the set of quintuples $\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right)$ such that:
(1) $k$ is the measure of $\gamma$, a common m-subsequence of some flattening of $A_{1} \ldots A_{i}$ and some flattening of $B_{1} \ldots B_{j}$, defined by (1).
(2) free m-set $F_{f} \subseteq_{m} A_{i}$ is the m-set of elements of $A_{i}$ which are not used by $\gamma$,
(3) free m-set $G_{f} \subseteq_{m} B_{j}$, is the m-set of elements of $B_{j}$ not used by $\gamma$,
(4) m-set of used elements $F_{u} \subseteq_{m} A_{i}$ is the m-set of elements of $A_{i}$ used by $\gamma$, and
(5) m-set of used elements $G_{u} \subseteq_{m} B_{j}$ is the m-set of elements of $B_{j}$ used by $\gamma$.

Example 3. $\quad(1.2,\{(c, 0.5),(a, 0.4)\},\{(b, 0.5)\},\{(a, 0.4)\},\{(d, 0.6),(e, 0.8)\})$ is in $\operatorname{Ent}(2,2)$ for m-set-strings in Example 2.
We refer to such quintuple as an entry. The measure of the CCS of the some flattening of $A_{1} \ldots A_{m}$ and some flattening of $B_{1} \ldots B_{n}$ is then, by definition, the largest $k$ such that $\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right) \in \operatorname{Ent}(m, n)$ for some m-sets $F_{f}, F_{u}, G_{f}$ and $G_{u} . \operatorname{Ent}(0,0)$ contains just one entry, namely $(0, \emptyset, \emptyset, \emptyset, \emptyset)$, while Ent $(i, j)$ can be computed dynamically from $\operatorname{Ent}(i-1, j)$ and $\operatorname{Ent}(i, j-1)$. The problem is that the cardinality of $\operatorname{Ent}(i, j)$ could become very large, making such an algorithm exponential in the worst case.
Let $e=\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right) \in \operatorname{Ent}(i-1, j)$ and $F^{s}, F_{u} \subseteq_{m} F^{s}$ be the m-set with the following property: $x \varepsilon F^{s} \Leftrightarrow x \varepsilon F_{u}$ and $\mu_{F_{u}}(x) \leq \mu_{F^{s}}(x)=\mu_{A_{i}}(x)$. It means, the m-set $F^{s}$ is the maximal m -set that has the same elements as the m-set $F_{u}$, but membership values of elements in $F^{s}$ are the same as in the m-set $A_{i}$.
Let $S$ be any subset of $A_{i} \cap_{m} G_{f}$. We say that $e$ vertically generates $e^{\prime} \in \operatorname{Ent}(i, j)$ iff

1. $e^{\prime}=\left(k+W(S)-W\left(A_{i} \cap_{m} G_{u}\right)+W\left(S^{\prime}\right), A_{i}-_{m} S, F_{u}, G_{f}-_{m} S, G_{u}\right)$ for any subset $S^{\prime}$ of $A_{i} \cap_{m} G^{s}, W\left(S^{\prime}\right)>W\left(A_{i} \cap_{m} G_{u}\right)$, or
2. $e^{\prime}=\left(k+W(S), A_{i}-_{m} S, F_{u}, G_{f}-_{m} S, G_{u}\right)$ and for each subset $S^{\prime}$ of $A_{i} \cap_{m} G^{s}$ is $W\left(S^{\prime}\right) \leq W\left(A_{i} \cap_{m} G_{u}\right)$.

The element $e^{\prime} \in \operatorname{Ent}(i, j)$ and it is shown by the following: If $\alpha$ is common msubsequence with a measure $k=\operatorname{Val}(\alpha)$ of the flattening of $A_{1} \ldots A_{i-1}$ and some flattening of $B_{1} \ldots B_{j}$, where $F_{f} \subseteq_{m} A_{i-1}$ and $G_{f} \subseteq_{m} B_{j}$ are free m-sets, and $\beta$ is a m-sequence consisting of the elements of $S \subseteq_{m} A_{i} \cap G_{f}$ written in any order, then $\alpha \beta$ (having measure $k+\operatorname{Val}(S)$ ) is common subsequence of a flattening of $A_{1} \ldots A_{i}$ and a flattening of $B_{1} \ldots B_{j}$, with free m-sets $A_{i}-S$ and $G_{f}-S$. The used elements from m-set $G_{u}$ can be used with some better membership values and it is evaluated by the comparison of the weights of the sets $A_{i} \cap_{m} G^{s}$ and $A_{i} \cap_{m} G_{u}$. If $W\left(A_{i} \cap_{m} G_{u}\right)<W\left(A_{i} \cap_{m} G^{s}\right)$ then there exists some better using of elements in $A_{i}$.

Similarly, if $\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right) \in \operatorname{Ent}(i, j-1)$ and $S \subseteq_{m} F_{f} \cap_{m} B_{j}$ and $S^{\prime}$ is any subset of $\left.B_{j} \cap_{m} F^{s}\right), W\left(S^{\prime}\right)>W\left(B_{j} \cap_{m} F_{u}\right)$, we say that ( $k, F_{f}, F_{u}, G_{f}, G_{u}$ ) horizontally generates $\left(k+W(S)-W\left(B_{j} \cap_{m} F_{u}\right)+W\left(S^{\prime}\right), F_{f}-_{m} S, F_{u}, B_{j}-_{m} S, G_{u}\right) \in \operatorname{Ent}(i, j)$ or $\left(k+W(S), F_{f}-{ }_{m} S, F_{u}, B_{j}-{ }_{m} S, G_{u}\right) \in \operatorname{Ent}(i, j)$ according to the relation $W\left(B_{j} \cap_{m}\right.$ $\left.F_{u}\right)</ \geq W\left(S^{\prime}\right)$.

Lemma 1 If $e \in \operatorname{Ent}(i, j)$ for $i+j \geq 0$ then $e$ is generated by some element of either $\operatorname{Ent}(i-1, j)$ or $\operatorname{Ent}(i, j-1)$.

Proof. $e=\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right) \in \operatorname{Ent}(i, j)$, it means $e=\alpha \beta, \beta$ is the part of elements in $A_{i} \cap_{m} B_{j}$. According to above construction, the part $\beta$ is the prolongation of some element $e^{\prime} \in \operatorname{Ent}(i-1, j)$ or $\operatorname{Ent}(i, j-1)$. In the part $\alpha$ should be elements with higher membership values.
The element $e \in \operatorname{Ent}(i, j)$ is generated from elements in $E(i-1, j)$ or $E n t(i, j-1)$ using of two sets: the free subset and the used subset of $B_{j}$, respectively $A_{i}$. The following algorithm is a dynamic programming algorithm in which the boundary conditions are set and then the interval entries are determined:

## Algorithm A.

for all $i$ do $\operatorname{Ent}(i, 0):=\left\{\left(0, A_{i}, \emptyset, \emptyset, \emptyset\right)\right\}$
for all $j$ do $\operatorname{Ent}(0, j):=\left\{\left(0, \emptyset, \emptyset, B_{j}, \emptyset\right)\right\}$
for $i:=1$ to $m$ do
for $j:=1$ to $n$ do
$\operatorname{Ent}(i, j):=\{$ all entries vertically generated from $\operatorname{Ent}(i-1, j)\}$
$\bigcup\{$ all entries horizontally generated from $\operatorname{Ent}(i, j-1)\}$
$\max \mathbf{k}:=$ the largest $k$ such that $\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right) \in \operatorname{Ent}(m, n)$ for some $F_{f}, F_{u}, G_{f}, G_{u}$.

### 3.2 Description of a better algorithm

The above algorithm may be very time-consuming because of too many quintuples is necessary to analyze. We will speed the algorithm by eliminating consideration of many quintuples.
If $\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right),\left(k^{\prime}, F_{f}^{\prime}, F_{u}^{\prime}, G_{f}^{\prime}, G_{u}^{\prime}\right) \in \operatorname{Ent}(i, j)$, we say that $\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right)$ dominates $\left(k^{\prime}, F_{f}^{\prime}, F_{u}^{\prime}, G_{f}^{\prime}, G_{u}^{\prime}\right)\left(\left(k^{\prime}, F_{f}^{\prime}, F_{u}^{\prime}, G_{f}^{\prime}, G_{u}^{\prime}\right) \preceq\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right)\right)$ if the following conditions hold:

1. $d=k-k^{\prime} \geq 0$,
2. $\left(W\left(F_{f}^{\prime}-F_{f}\right) \leq d\right.$ and $\left.F_{u}^{\prime} \subseteq_{m} F_{u}\right)$ or $\left(W\left(F_{u}^{\prime}-F_{u}\right) \leq d\right.$ and $\left.F_{f}^{\prime} \subseteq_{m} F_{f}\right)$,
3. $\left(W\left(G_{f}^{\prime}-G_{f}\right) \leq d\right.$ and $\left.G_{u}^{\prime} \subseteq_{m} G_{u}\right)$ or $\left(W\left(G_{u}^{\prime}-G_{u}\right) \leq d\right.$ and $\left.G_{f}^{\prime} \subseteq_{m} G_{f}\right)$.

The relation " $\preceq$ " is a transitive, antisymmetric and reflexive relation. The elements of $E n t(i, j)$ can be ordered according to relation " $\preceq$ ", it means they are ordered in some chains. The last element of the chain has maximal measure and that is very important.

Lemma 2 Any element of Ent $(i, j)$ which is not maximal with respect to the relation " $\preceq$ " can be discarded during execution of the algorithm without affecting the final value of max_k.

Proof. It will be proved by downward induction on both indices $i$ and $j$. The value of max_k is obtained from $\operatorname{Ent}(m, n)$ in the last step and all other elements may be discarded with no effect.
Suppose $i+j<m+n$ and $e^{\prime} \in \operatorname{Ent}(i, j), e^{\prime}$ is not maximal. Let $e \in \operatorname{Ent}(i, j)$ is maximal. It means, $e^{\prime} \preceq e$. It is necessary to prove that maximal element of $\operatorname{Ent}(i+1, j)$ or $\operatorname{Ent}(i, j+1)$ which is generated by $e^{\prime}$ can be generated by $e$ too. And the element $e^{\prime}$ can be discarded.
Let $e=\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right), e^{\prime}=\left(k^{\prime}, F_{f}^{\prime}, F_{u}^{\prime}, G_{f}^{\prime}, G_{u}^{\prime}\right)$ and $e^{\prime}$ vertically generates $f^{\prime} . f^{\prime}$ should have two forms for some m-set $P \subseteq A_{i+1} \cap G_{f}^{\prime}$
(a) $f^{\prime}=\left(k^{\prime}+W(P)-W\left(A_{i+1} \cap_{m} G_{u}^{\prime}\right)+W\left(A_{i+1} \cap_{m} G^{\prime s}\right), A_{i+1}-{ }_{m} P, P, G_{f}^{\prime}-{ }_{m} P, G_{u}^{\prime}\right)$, if $W\left(A_{i+1} \cap_{m} G_{u}^{\prime}\right)<W\left(A_{i+1} \cap_{m} G^{\prime s}\right)$, or
(b) $f^{\prime}=\left(k^{\prime}+W(P), A_{i+1}-{ }_{m} P, P, G_{f}^{\prime}-{ }_{m} P, G_{u}^{\prime}\right)$, if $W\left(A_{i+1} \cap_{m} G_{u}^{\prime}\right) \geq W\left(A_{i+1} \cap_{m} G^{\prime s}\right)$.

Let $S=P \cap G_{f}$, and $f$ is vertically generated by $e$. $f$ should have two forms: (1) $f=\left(k+W(S)-W\left(A_{i+1} \cap_{m} G_{u}\right)+W\left(A_{i+1} \cap_{m} G^{s}\right), A_{i+1}-{ }_{m} S, S, G_{f}-{ }_{m} S, G_{u}\right)$ or (2) $f=\left(k+W(S), A_{i+1}-{ }_{m} S, S, G_{f}-{ }_{m} S, G_{u}\right)$. It is necessary to analyze four cases to prove the Lemma (a)-(1), (a)-(2), (b)-(1), (b)-(2). We start with the first one, it means (a)-(1), and $W\left(A_{i+1} \cap_{m} G_{u}^{\prime}\right)<W\left(A_{i+1} \cap_{m} G^{\prime s}\right)$ and $W\left(A_{i+1} \cap_{m} G_{u}\right)<W\left(A_{i+1} \cap_{m} G^{s}\right)$.

Since $e^{\prime} \preceq e, d=k-k^{\prime}$,
$d \geq 0,\left(\overline{(W}\left(F_{f}^{\prime}-F_{f}\right) \leq d\right.$ and $\left.F_{u}^{\prime} \subseteq_{m} F_{u}\right)$ or $\left(W\left(F_{u}^{\prime}-F_{u}\right) \leq d\right.$ and $\left.\left.F_{f}^{\prime} \subseteq_{m} F_{f}\right)\right)$, and $\left(\left(W\left(G_{f}^{\prime}-G_{f}\right) \leq d\right.\right.$ and $\left.G_{u}^{\prime} \subseteq_{m} G_{u}\right)$ or $\left(W\left(G_{u}^{\prime}-G_{u}\right) \leq d\right.$ and $\left.\left.G_{f}^{\prime} \subseteq_{m} G_{f}\right)\right)$. Then $W\left(P-_{m} S\right)=W\left(P-_{m} P \cap_{m} G_{f}\right)=W\left(P-_{m} G_{f}\right) \leq W\left(F_{f}^{\prime}-_{m} F_{f}\right) \leq d$ and $W\left(P-_{m} S\right) \leq W(P)-W(S)$. Let $d^{\prime}=(W(P)-W(S))-\left(W\left(A_{i+1} \cap_{m} G^{s}\right)-\right.$ $\left.W\left(A_{i+1} \cap_{m} G^{\prime s}\right)\right)-\left(W\left(A_{i+1} \cap_{m} G_{u}\right)-W\left(A_{i+1} \cap_{m} G_{u}^{\prime}\right) \leq d\right.$ We prove that $f^{\prime} \preceq f$, it means $f^{\prime}$ is not maximal or $f=f^{\prime}$. According to definition of " $\preceq$ " it is necessary to check three conditions 1.-3.

1. $z=k+W(S)-W\left(A_{i+1} \cap_{m} G_{u}\right)+W\left(A_{i+1} \cap_{m} G^{s}\right)-\left(k^{\prime}+W(P)-W\left(A_{i+1} \cap_{m}\right.\right.$ $\left.\left.G_{u}^{\prime}\right)+W\left(A_{i+1} \cap_{m} G^{\prime s}\right)\right)=k-k^{\prime}-(W(P)-W(S))+W\left(A_{i+1} \cap_{m} G^{s}\right)-W\left(A_{i+1} \cap_{m}\right.$ $\left.G^{\prime s}\right)+W\left(A_{i+1} \cap_{m} G_{u}\right)-W\left(A_{i+1} \cap_{m} G_{u}^{\prime} \geq d-d^{\prime} \geq 0\right.$
2. $W\left(P-{ }_{m} S\right) \leq d$ and $A_{i+1}-_{m} P \subseteq_{m} A_{i+1}-{ }_{m} S$
3. $W\left(G_{f}^{\prime}-{ }_{m} P-{ }_{m}\left(F_{f}-_{m} S\right)\right)=W\left(G_{f}^{\prime}-{ }_{m} G_{f}\right) \leq d$ and $G_{u}^{\prime} \subseteq_{m} G_{u}$.

The rest three cases can be proved by a very similar method. And the vertical case is similar.

If $e=\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right) \in \operatorname{Ent}(i, j)$, we define the horizontal child of $e$ to be hor $(e)=$ $\left.k+W\left(F_{f} \cap B_{j+1}\right)-W\left(A_{i} \cap_{m} G_{u}\right)+W\left(A_{i} \cap_{m} G^{s}\right), F_{f}-B_{j+1}, F_{u}, B_{j+1}-F_{f}, G_{u}\right)$ or $\left.\operatorname{hor}(e)=k+W\left(F_{f} \cap B_{j+1}\right), F_{f}-B_{j+1}, F_{u}, B_{j+1}-F_{f}, G_{u}\right)$ and define the vertical child of $e$ to be $\operatorname{ver}(e)=k+W\left(A_{i+1} \cap G_{f}\right)-W\left(B_{j+1} \cap G_{u}\right)+W\left(B_{j+1} \cap G^{s}\right), B_{j+1}-$ $\left.G_{f}, F_{u}, G_{f}-B_{j+1}, G_{u}\right)$ or $\left.\operatorname{ver}(e)=k+W\left(A_{i+1} \cap G_{f}\right), B_{j+1}-G_{f}, F_{u}, G_{f}-B_{j+1}, G_{u}\right)$. We define MaxEnt $(i, j)$ to be the set of maximal elements of $\operatorname{Ent}(i, j)$ under the dominance relation " $\preceq$ ".

Lemma 3 Any entry horizontally generates at most one maximal entry and vertically generates at most one maximal entry.

Proof. Let $e=\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right) \in \operatorname{Ent}(i, j)$. The only elements vertically generated by $e$ which can be maximal are in the $\operatorname{ver}(e)$, since they dominates any others vertically generated by $e$. Similarly, hor $(e)$ dominates any entries horizontally generated by $e$.

We say that $\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right)$ strongly dominates $\left(k^{\prime}, F_{f}, F_{u}, G_{f}, G_{u}\right)$ if $k>k^{\prime}$. If $S \subseteq \operatorname{Ent}(i, j)$, defines $\operatorname{Dom}(S) \subseteq S$ to be the set obtained by deleting every element of $S$ which is strongly dominated by another element of $S$. We now inductively define sets $\operatorname{Chain}(i, j) \subseteq \operatorname{Ent}(i, j)$ by:

1. $\operatorname{Chain}(i, 0)=\left\{\left(0, A_{i}, \emptyset, \emptyset, \emptyset\right)\right\}$,
2. $\operatorname{Chain}(0, j)=\left\{\left(0, \emptyset, \emptyset, B_{j}, \emptyset\right)\right\}$,
3. $\operatorname{Chain}(i, j)=\operatorname{Dom}(\{\operatorname{ver}(e) \mid e \in \operatorname{Chain}(i-1, j)\} \cup\{\operatorname{hor}(e) \mid e \in \operatorname{Chain}(i, j-1)\})$.

We refer to entries Chain $(i, j)$ as weakly maximal. We observe the following lemma.
Theorem $1 \operatorname{MaxEnt}(i, j) \subseteq \operatorname{Chain}(i, j)$.

Proof. By induction. For $i=0$ or $j=0$ the two sets $\operatorname{MaxEnt}(i, j)$ and $\operatorname{Chain}(i, j)$ are identical. For $i, j>0$, and $e \in \operatorname{MaxEnt}(i, j)$ must be vertical or horizontal child of some maximal element, which is weakly maximal by induction. It means, $e$ must be weakly maximal, since it is maximal and thus cannot be deleted by operator Dom.

Using the results of the Lemmas 2 and 3 and Theorem 1 we have the following algorithm:

## Algorithm B.

\{Using weakly maximal entries.\}
for all i do $\operatorname{Chain}(i, 0):=\left\{\left(0, A_{i}, \emptyset, \emptyset, \emptyset\right)\right\}$;
for all $\mathbf{j}$ do $\operatorname{Chain}(0, j):=\left\{\left(0, \emptyset, \emptyset, B_{j}, \emptyset\right)\right\} ;$
for $\mathrm{i}:=1$ to m do
for $\mathrm{j}:=1$ to n do

## begin

Chain $(i, j):=\emptyset$;
for all $\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right) \in \operatorname{Chain}(i, j-1)$ do begin
help $:=W\left(B_{j} \cap_{m} F^{s}\right)-W\left(B_{j} \cap_{m} F_{u}\right) ;$
if help $\leq 0$ then
insert $\left(k+W\left(F \cap B_{j}\right), F_{f}-B_{j}, F_{u}, B_{j}-F_{f}, G_{u}\right)$ into Chain $(i, j)$
else insert $\left(k+W\left(F_{f} \cap B_{j}\right)+\right.$ help, $\left.F_{f}-B_{j}, F_{u}, B_{j}-F_{f}, G_{u}\right)$ into Chain $(i, j)$ end;
for all $\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right) \in \operatorname{Chain}(i-1, j)$ do begin
help $:=W\left(A_{i} \cap_{m} G^{s}\right)-W\left(A_{i} \cap_{m} G_{u}\right) ;$
if help $\leq 0$ then
insert $\left(k+W\left(A_{i} \cap G\right), A_{i}-G_{f}, F_{u}, G_{f}-A_{i}, G_{u}\right)$ into Chain $(i, j)$
else insert $\left(k+W\left(G_{f} \cap A_{i}\right)+\right.$ help, $\left.A_{i}-G_{f}, F_{u}, G_{f}-A_{i}, G_{u}\right)$ into Chain $(i, j)$
end;
delete all nonweakly maximal elements from $\operatorname{Chain}(i, j)$
end
$\max \_k:=$ the maximum value of $k$ such that $\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right) \in \operatorname{Chain}(m, n)$ for some $F_{f}, F_{u}, G_{f}$ and $G_{u}$.
The algorithm works in $O(m \cdot n \cdot K \cdot t)$-time, where $K$ is the maximal number of elements in Chain $(i, j)$ and $t$ is the maximal time spent for computing the intersection of two sets. The algorithm works in $O(m \cdot n \cdot k)$-space, where $k$ is the maximal number of elements in the m-sets $A_{i}, B_{j}$. In the next section we show the idea of some efficient implementation of the algorithm.

### 3.3 Efficient implementation of algorithm B.

We show the structure of $\operatorname{Chain}(i, j)$ that will help obtain an efficient implementation of algorithm B. We begin by defining a transitive reflexive relation $\triangleleft$ on $\operatorname{Ent}(i, j)$. We say that $\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right) \triangleleft\left(k^{\prime}, F_{f}^{\prime}, F_{u}^{\prime}, G_{f}^{\prime}, G_{u}^{\prime}\right)$ if $F_{f} \subseteq_{m} F_{f}^{\prime}, G_{f}^{\prime} \subseteq_{m} G_{f}, F_{u}={ }_{m} F_{u}^{\prime}$ and $G_{u}={ }_{m} G_{u}^{\prime}$.

Lemma 4 (a) If $e, e^{\prime} \in \operatorname{Ent}(i-1, j)$, and if $e \triangleleft e^{\prime}$, then $\operatorname{ver}(e) \triangleleft \operatorname{ver}\left(e^{\prime}\right)$.
(b) If $e, e^{\prime} \in \operatorname{Ent}(i, j-1)$, and if $e \triangleleft e^{\prime}$, then hor $(e) \triangleleft \operatorname{hor}\left(e^{\prime}\right)$.
(c) If $e \in \operatorname{Ent}(i, j-1)$ and $e^{\prime} \in \operatorname{Ent}(i-1, j)$, then $\operatorname{hor}(e) \triangleleft \operatorname{ver}\left(e^{\prime}\right)$.

Proof. (a) ver $(e)=\left(k, F_{f} \cap_{m} B_{j}, F_{u}, B_{j}-{ }_{m} F_{f}, G_{u}\right)$ and $\operatorname{ver}\left(e^{\prime}\right)=\left(k^{\prime}, F^{\prime}-_{m}\right.$ $\left.B_{j}, F_{u}^{\prime}, B_{j}-{ }_{m} F_{f}^{\prime}, G_{u}^{\prime}\right)$. It follows from $F_{f} \subseteq_{m} F_{f}^{\prime}$ that $F_{f}-B_{j} \subseteq_{m} F_{f}^{\prime}-B_{j}$ and $B_{j}-F_{f}^{\prime} \subseteq_{m} B_{j}-F_{f}$, i.e. $\operatorname{ver}(e) \triangleleft \operatorname{ver}\left(e^{\prime}\right)$.
(b) Similar to the proof of (a).
(c) Let $e=\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right)$ and $e^{\prime}=\left(k^{\prime}, F_{f}^{\prime}, F_{u}^{\prime}, G_{f}^{\prime}, G_{u}^{\prime}\right)$. Then hor $(e)=\left(k, F_{f}-{ }_{m}\right.$ $\left.B_{j}, F_{u}, B_{j}-{ }_{m} F_{f}, G_{u}\right)$ and $\operatorname{ver}\left(e^{\prime}\right)=\left(k^{\prime}, F_{f}^{\prime}-{ }_{m} B_{j}, F_{u}^{\prime}, B_{j}-{ }_{m} F_{f}^{\prime}, G_{u}^{\prime}\right)$. It can be seen that $F_{f} \subseteq_{m} A_{i}$ since $e \in \operatorname{Ent}(i, j-1)$, and that $G_{f}^{\prime} \subseteq_{m} B_{j}$ since $e^{\prime} \in \operatorname{Ent}(i-1, j)$. And we have $F_{f}-B_{j} \subseteq_{m} A_{i}-G_{f}^{\prime}$ and $G_{f}^{\prime}-A_{i} \subseteq_{m} B_{j}-F_{f}$, i.e. $h o r(e) \triangleleft \operatorname{ver}\left(e^{\prime}\right)$.

Lemma 5 The relation $\triangleleft$ imposes a total ordering on Chain $(i, j)$.
Proof. We need to prove that for any distinct $f, f^{\prime} \in \operatorname{Chain}(i, j)$, either $f \triangleleft f^{\prime}$ or $f^{\prime} \triangleleft f$ but not both. If $f^{\prime} \triangleleft f$ and $f \triangleleft f^{\prime}$, then $f$ and $f^{\prime}$ would have the same free sets, which implies they must be identical, else the one with the smaller value of $k$ would not be weakly maximal. Thus, we need only show that $f$ and $f^{\prime}$ are comparable. We do this by induction on $i$ and $j$.
$\operatorname{Chain}(0, j)$ contains just one entry, namely $\left(0, \emptyset, \emptyset, B_{j}, \emptyset\right)$, and hence is ordered. Similarly, $\operatorname{Chain}(i, 0)$ contains only the entry $\left(0, A_{i}, \emptyset, \emptyset, \emptyset\right)$.
Suppose, $i, j>0$, and $f, f^{\prime} \in \operatorname{Chain}(i, j)$. Both $f$ and $f^{\prime}$ must be generated by maximal entries $e$ and $e^{\prime}$, respectively. We consider three cases. If $f$ and $f^{\prime}$ are the vertical children of $e$ and $e^{\prime}$, respectively, then by induction, $e$ and $e^{\prime}$ are comparable, hence $f$ and $f^{\prime}$ are comparable by above Lemma. If $f$ and $f^{\prime}$ are the horizontal children of $e$ and $e^{\prime}$ the proof is similar. If $f$ is the horizontal child of $e$ and $f^{\prime}$ is vertical child of $e^{\prime}$, then $f$ and $f^{\prime}$ are comparable by above Lemma too.

Lemma 6 Chain $(i, j)$ has the number of elements at most $1+\left|A_{i}\right|+\left|B_{j}\right|$, where $\left|A_{i}\right|,\left|B_{j}\right|$ are the numbers of elements in the $m$-sets $A_{i}, B_{j}$, for $i=1, \ldots, m, j=$ $1, \ldots, n$.

Proof. The main idea of the proof is in the following: Each element from m-set should be used once at most but with some different membership value.
If $e \in\left(k, F_{f}, F_{u}, G_{f}, G_{u}\right) \in \operatorname{Ent}(i, j)$, define signature of $e$ to be $\left|F_{f}\right|-\left|G_{f}\right|$, which must lie in the interval $\left[-\left|B_{j}\right| . .\left|A_{i}\right|\right]$. Since $\operatorname{Chain}(i, j)$ is ordered under the relation $\triangleleft$, each entry must have different signature.
It means, the algorithm works in $O(m \cdot n \cdot L \cdot t)$-time, where $L$ is the maximal number of the numbers in $\left\{1+\left|A_{i}\right|+\left|B_{j}\right|, i=1, \ldots m, j=1, \ldots n\right\}$ and $t=\max \left\{\left|A_{i}\right|,\left|B_{j}\right|, i=\right.$ $1, \ldots m, j=1, \ldots n\}$ is the maximal time spent for computing of the intersection of two sets. The algorithm works in $O(n \cdot L \cdot t)$-space.

## 4 CONCLUDING REMARKS

Polynomial algorithms for the solutions of the SSCCS and MSSCCS Problem with membership functions have been presented. The algorithms work in $O(m \cdot n \cdot L \cdot t)$-time and $O(n \cdot L \cdot t)$-space, where $L=\max \left\{1+\left|A_{i}\right|+\left|B_{j}\right|, i=1, \ldots m, j=1, \ldots n\right\}$ and $t=\max \left\{\left|A_{i}\right|,\left|B_{j}\right|, i=1, \ldots m, j=1, \ldots n\right\}$ is the maximal time spent for computing of the intersection of two sets.

## References

[1] Andrejková, G.: The longest restricted common subsequence problem. Proceedings of the Prague Stringology Club Workshop'98, Prague, 1998, p. 14-25.
[2] Andrejková, G.: The Set Closest Common Subsequence Problem, Proceedings of 4-th International Conference on Applied Informatics '99, Eger-Noszvaj, (1999), p. 8 .
[3] Gottwald, S.: Fuzzy Sets and Fuzzy Logic, Vieweg, Wiesbaden, (1993), p. 216.
[4] Heckel, P.: A technique for isolating differences between files. Comm. ACM 21, 4 (Apr. 1978), p. 264-268.
[5] Hirschberg, D. S.: Algorithms for longest common subsequence problem. Journal ACM 24, 4 (Oct 1977), p. 664-675.
[6] Hirschberg, D. S., Larmore, L. L.: Set-Set LCS Problem. Algorithmica 4 (1989), p. 503-510.
[7] Holub, J.: Dynamic Programming for Reduced NFAs for Approximate String and Sequence Matching. Proceedings of the Prague Stringology Club Workshop'98, Prague, 1998, p. 73-82.
[8] Lowrance, R., Wagner, R. A.: An extension of the string-to-string correction problems. Journal ACM 22, 2 (Apr. 1975), p. 177-183.
[9] Mohanty, S. P.: Shortest string containing all permutations. Discrete Mathematics 31, 1980, p. 91-95.
[10] Needleman, S. B., Wunsch, Ch. D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal Mol. Biol. 48, 1970, p. 443-453.
[11] Troníček, Z., Melichar, B.: Directed Acyclic Subsequence Graph. Proceedings of the Prague Stringology Club Workshop'98, Prague, 1998, p. 107-118.
[12] Zadeh, L.: Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets Syst. 1, 1978, p. 3-28.


[^0]:    ${ }^{1}$ This research was partially supported by Slovak Grant Agency for Science VEGA, project No. 1/4375/97

