Regular Expression Constrained Sequence Alignment Revisited*

Gregory Kucherov[1]
Tamar Pinhas[2]
Michal Ziv-Ukelson[2]

(1) LIGM/CNRS, Marne-la-Vallée
(2) Ben-Gurion University, Israel

* appeared in IWOCA'2010

- Sequence alignment

(Pairwise) sequence alignment

Sequences producing significant alignments: Score (bits) E value

gnl|Pfam|pfam00155 aminotran_1, Aminotransferases class-I 338 4e-94

gnl|Pfam|pfam00155, aminotran_1, Aminotransferases class-I

Add query to multiple alignment, display up to 10 sequences most similar to the query

```
            Length = 428
            Score =  338 bits (857), Expect = 4e-94

Query:  23   KSTWFSEVQMGPPDAILGVTEAFKKDTNPKKIN----LGAGAYRDDNTQPFVLPSVREAE  78
Sbjct:  1    LSRNATFNSHGQDSSYFLGWQEYEKNPYHEVHNTNGIIQMGLAENQLCFDLLESWLAKNP  60

Query:  79   KRVVSRS-------LDKEYATIIGIPEFYNKAIELALGKGSKRLAAKHNVTAQSISGTGA  131
Sbjct:  61   EAAAFKKNGESIFAELALFQDYHGLPAFKKAMVDFMAEIRGNKVTFDPNHLVLTAGATSA  120

Query:  132  LRIGAAFLAKFWQGNREIYIPSPSWGNHV-AIFEHAGLPVNRYRYYDKDTCALDFGGLIE  190
Sbjct:  121  NETFIFCLADPGE---AVLIPTPYYPGFDRDLKWRTGVEIVPIHCTSSNGFQITETALEE  177

Query:  191  DLKKIPE---KSIVLLHACAHNPTGVDPTLEQWREISALVKKRNLYPFIDMAYQGFATGD  247
Sbjct:  178  AYQEAEKRNLRVKGVLVTNPSNPLGTTMTRNELYLLLSFVEDKGIHLISDEIYSGTAFSS  237

Query:  248  IDRDAQAVRTFEAD---------GHDFCLAQSFAKNMGLYGERAGAFTVLCSDEEEAARV  298
Sbjct:  238  P--SFISVMEVLKDRNCDENSEVWQRVHVVYSLSKDLGLPGFRVGAIYSNDDMVVAAATK  295

Query:  299  M-------SQVKILIRGLYSNP---PVHGARIAAEILNNEDLRAQWLKDVKLMADRIIDV  348
Sbjct:  296  MSSFGLVSSQTQHLLSAMLSDKKLTKNYIAENHKRLKQRQKKLVSGLQKSG-ISCLNGNA  354

Query:  349  RTKLKDNLIKLGSSQNWDHIVNQIGMFCFTGLKPEQVQK-LIKDHSVYLTNDGRVSMAGV  407
Sbjct:  355  GLFCWVDMRHLLR----SNTFEAEMELWKKIVYEVHLNISPGSSCHCTEPGWFRVCFANL  410

Query:  408  TSKNVEYLAESIHKVTK  424
Sbjct:  411  PERTLDLAMQRLKAFVG  427
```

Image from: http://www.ncbi.nlm.nih.gov/

(Pairwise) sequence alignment

Consider an alphabet Σ and let $\Sigma' = \Sigma \cup \{-\}$.

Let s: $(\Sigma' \times \Sigma') \setminus \{(-,-)\} \rightarrow \Re$ be a scoring function.

Given two strings $a, b \in \Sigma^*$, an alignment of a and b is a pair $A, B \in (\Sigma')^*$ such that

- $|A| = |B|$
- $\forall i$, $(A_i, B_i) \neq (-,-)$
- deleting all' –' from A (resp. B) yields a (resp. b)

Optimal alignment maximizes score $s(A,B) = \sum s(A_i, B_i)$

(Pairwise) sequence alignment

Example: alignment of two protein sequences

```
NLGP--KDFGKISE-REFDNK
 | |    ||||    ||  |
QLNPLERSFGKINMRLEFA-K
```

Optimal alignement of two sequences $a[1..n]$ and $b[1..n]$ is computed in time $O(n^2)$ by Dynamic Programming: compute maximal score $T_{i,j}$ between $a[1..i]$ and b[$1..j$]

- Sequence alignment
- RECSA

Regular Expression Constrained Sequence Alignment (RECSA) problem:

Given two sequences and a regular language specified by a NFA *A*, find an optimal alignment that contains aligned substrings, each belonging to L(*A*).

- Sequence alignment
- RECSA

Example: TGFPSVGKTKDDA vs TFSVAKDDDGKSA

```
T G F P S V G K T K D D - - - - A
|   |   | |       | | |         |
T - F - S V A - - K D D D G K S A
```

The P-loop motif $(G+A)\Sigma\Sigma\Sigma\Sigma GK(S+T)$

```
T - - - G F P S V G K T K D D A
|           |   |         |
T F S V A K D D D G K S - - - A
      * * * * * * * *
```

Image from: Chung et al., Efficient algorithms for regular expression constrained sequence alignment, 2006

- Sequence alignment
- RECSA
- Existing results

Arslan's algorithm for RECSA problem

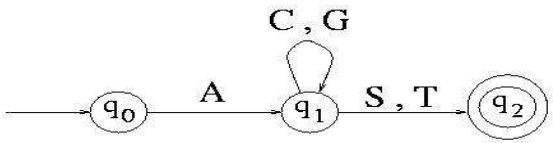
[Arslan CPM05]: A - an ε-free NFA

- Create an NFA M as follows:
  - states of M are ordered pairs of states of A
  - alphabet of M is $\Sigma_M=(\Sigma\cup\{-\})\times(\Sigma\cup\{-\})\setminus(-,-)$
  - if $q_1 \to^x p_1$ and $q_2 \to^y p_2$ are transitions of A, then $(q_1,q_2) \to^{(x,y)} (p_1,p_2)$, $(q_1,q_2) \to^{(-,y)} (p_1,p_2)$, $(q_1,q_2) \to^{(x,-)} (p_1,p_2)$ are transitions of M
  - add transitions $(q_0,q_0) \to (q_0,q_0)$ and $(q_f,q_f) \to (q_f,q_f)$ by all symbols of Σ_M
- $L(M)$ = all alignments containing a segment in which both aligned substrings belong to $L(A)$.

- Sequence alignment
- RECSA
- Existing results
-
-

Arslan's algorithm - example

A



M

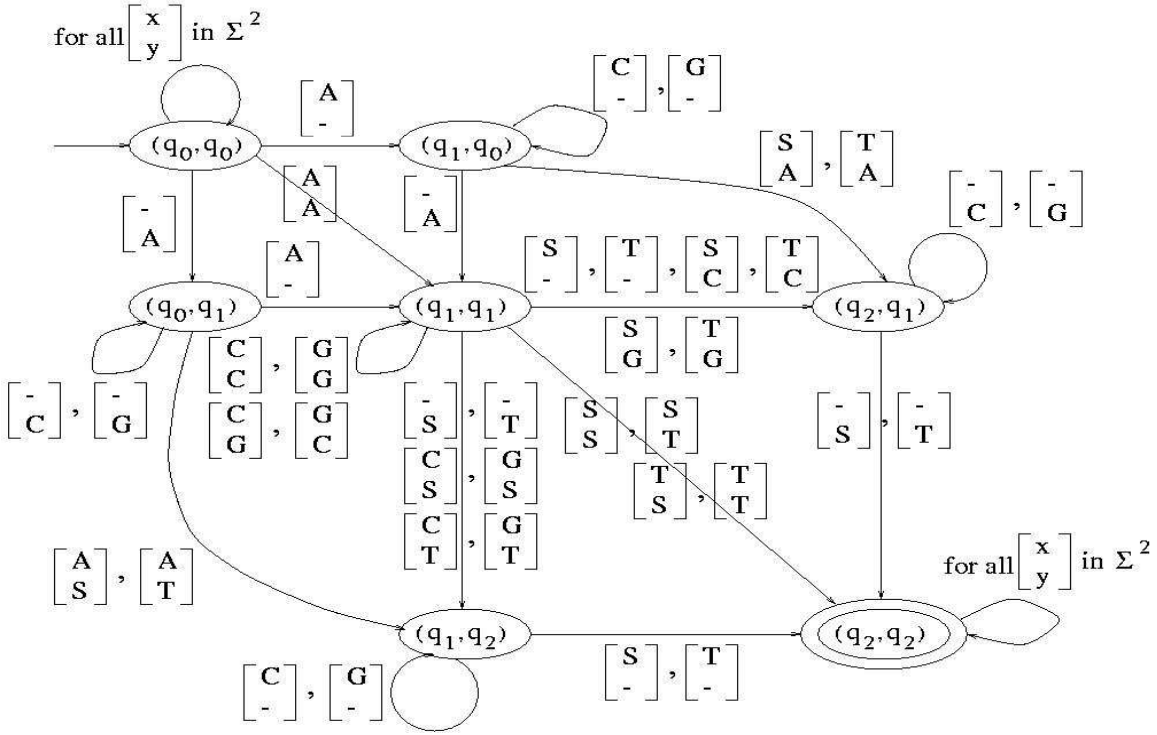


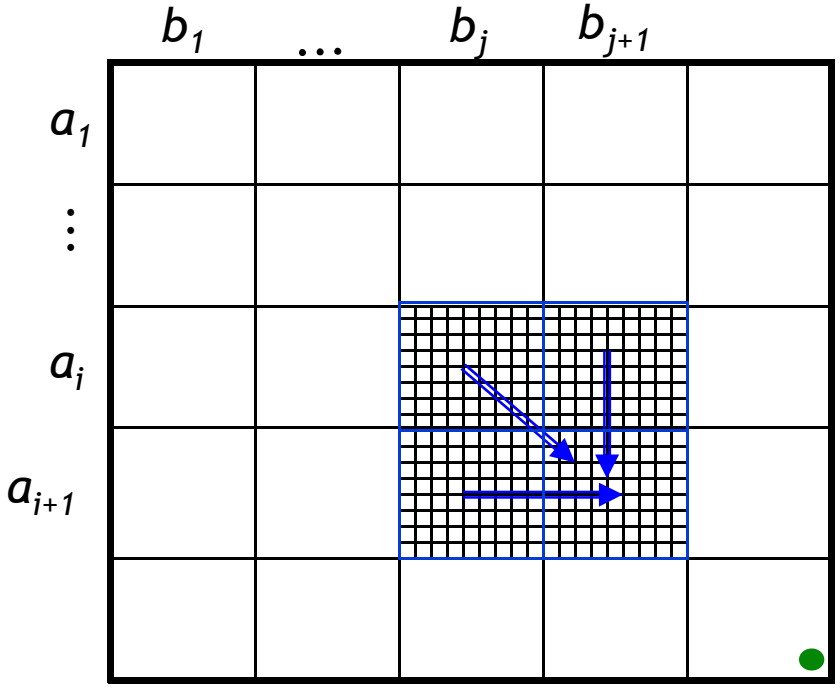
Image from: Arslan, Regular Expression Constrained Sequence Alignment, 2005

- Sequence alignment
- RECSA
- Existing results
-
-

Arslan's algorithm for RECSA problem

Input: NFA A, strings $a[1..n]$, $b[1..n]$, scoring $s(\cdot,\cdot)$

- *Dynamic Programming*: $T_{i,j}(q,p)$ is the maximal score of an alignment of $a[1..i]$ and $b[1..j]$ such that reading it by M leads to (q,p)

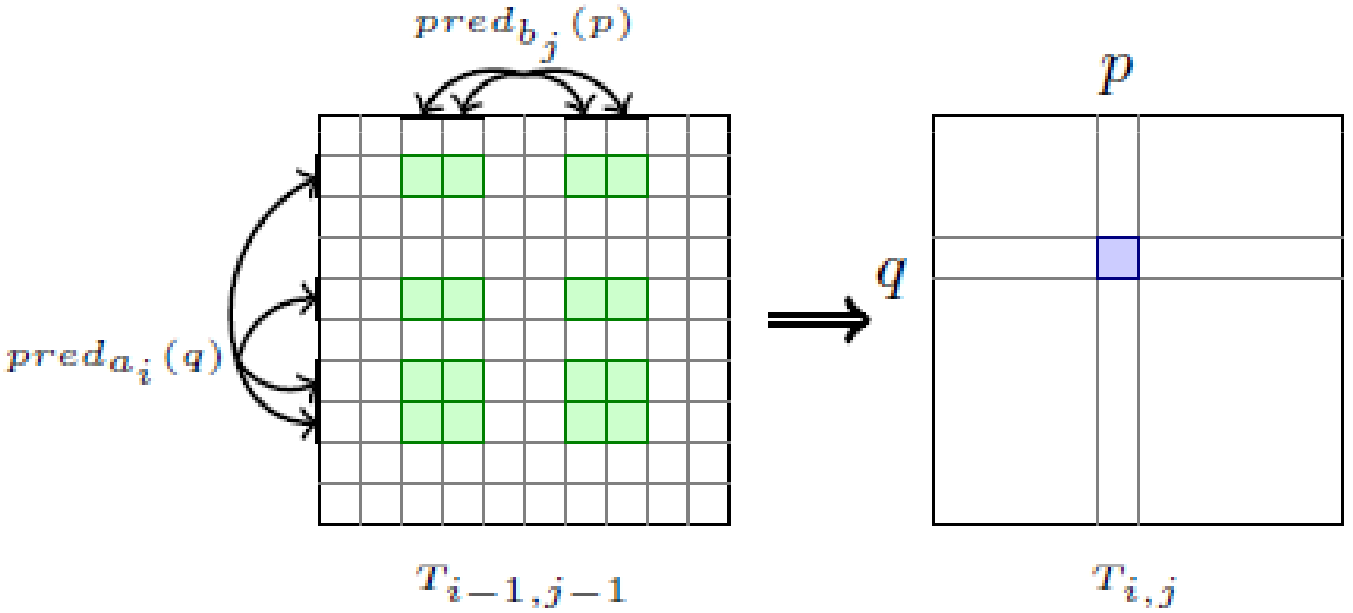


- Optimal score is max $\{ T_{|a|,|b|}(q_f,p_f) \mid q_f,p_f \in F_A \}$

- Sequence alignment
- RECSA
- Existing results
-
-

Arslan's algorithm cont.

$$T_{i,j}(q,p) = \max \begin{cases} \max\{T_{i-1,j}(q',p) | q' \in pred_{a_i}(q)\} + s(a_i,-) \\ \max\{T_{i,j-1}(q,p') | p' \in pred_{b_j}(p)\} + s(-,b_j) \\ \max\{T_{i-1,j-1}(q',p') | q' \in pred_{a_i}(q),\ p' \in pred_{b_j}(p)\} + s(a_i,b_j) \end{cases} (*)$$



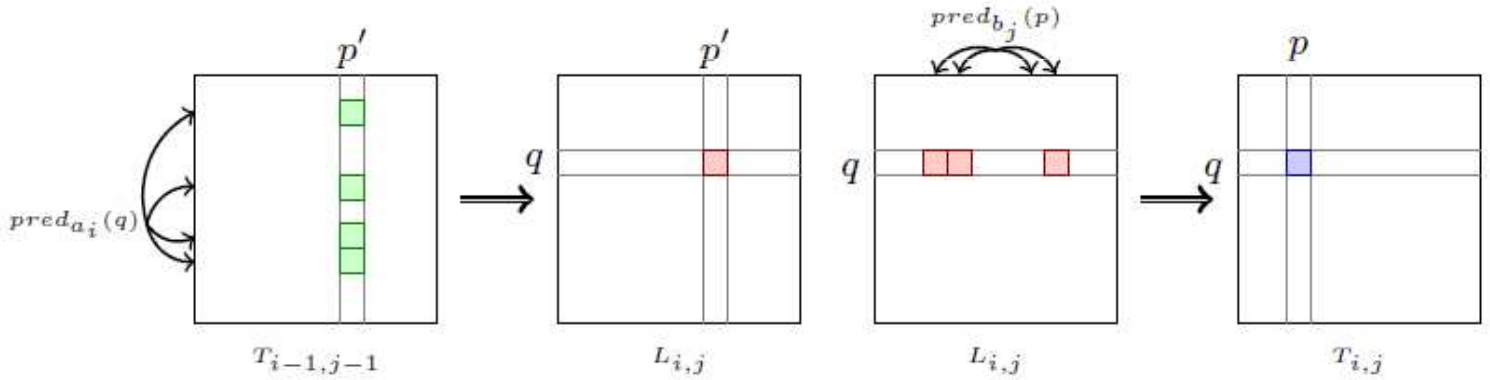
complexity: $O(n^2 \cdot |\delta|^2) = O(n^2 \cdot t^4)$ time and $O(n^2 \cdot t^2)$ space
(t nb of states and $|\delta|$ nb of transitions in the automaton)

- Sequence alignment
- RECSA
- Existing results
-
-

Chung et al's improvement

[Chung,Lu,Tang CPM06]: Split the computation into two steps using an auxiliary table, *L*

$$L_{i,j}(q,p') = \max\{T_{i-1,j-1}(q',p') \mid q' \in pred_{a_i}(q)\}$$
$$T_{i,j}(q,p) = \max\{L_{i,j}(q,p') \mid p' \in pred_{b_j}(p)\} + s(a_i,b_j)$$



(a) Calculation of $L_{i,j}$ (b) Calculation of $T_{i,j}$

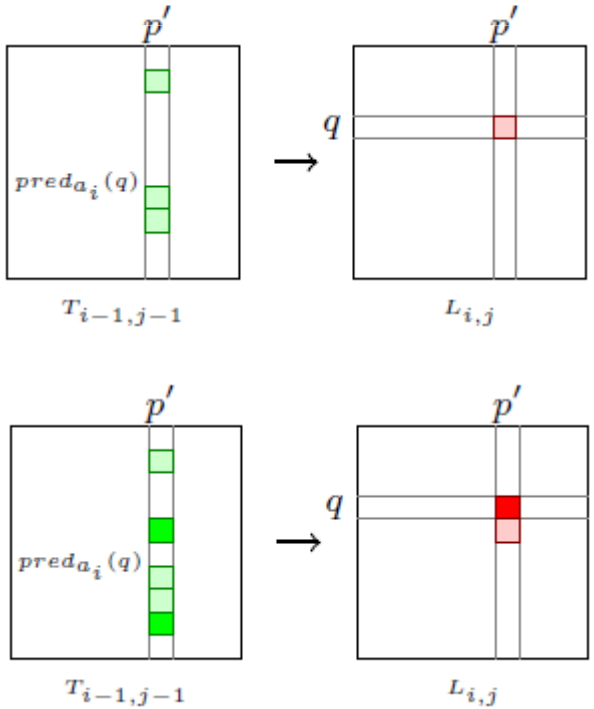
complexity: $O(n^2 \cdot t \cdot |\delta|) = O(n^2 \cdot t^3)$ time and $O(n^2 t^2)$ space

- Sequence alignment
- RECSA
- Existing results
- Improved solution
-

RECSA: a faster algorithm.

Observation: Chung et. al.'s algorithm may contain duplicate score maxima calculations

$$L_{i,j}(q,p') = \max\{T_{i-1,j-1}(q',p') | q' \in pred_{a_i}(q)\}$$
$$T_{i,j}(q,p) = \max\{L_{i,j}(q,p') | p' \in pred_{b_j}(p)\} + s(a_i,b_j)$$



- Sequence alignment
- RECSA
- Existing results
- Improved solution
-

RECSA: a faster algorithm.

Idea: Eliminate duplicate computations of maxima values over intersecting subsets of scores

THE SUBSET MAXIMA PROBLEM:

Let $W=\{x_1,\ldots,x_t\}$ be a set of weights, with and let $v_1,\ldots,v_t \in 2^W$. Compute $max(v_k)$ for each v_k using as few *max* operations as possible.

- Sequence alignment
- RECSA
- Existing results
- Improved solution
-

IDEA 1: Steiner tree strategy

Start with empty set and compute maximum of each set *element-by-element*, sharing common computations as much as possible

- Sequence alignment
- RECSA
- Existing results
- Improved solution
-

IDEA 1: Steiner tree strategy

Start with empty set and compute maximum of each set *element-by-element*, sharing common computations as much as possible

sets of $2^W \equiv$ boolean vectors of $\{0,1\}^t$

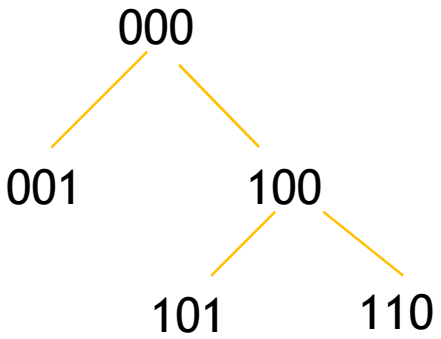
max between a current set and an element ≡ replacing 0 by 1 in the vector

$\Rightarrow$ Steiner tree in the Hamming hypercube

- Sequence alignment
- RECSA
- Existing results
- Improved solution
-

IDEA 1: Steiner tree strategy

MINIMAL STEINER TREE PROBLEM ON HAMMING HYPERCUBE: Given a set *S* of *t* points in *t*-dimensional Hamming hypercube, find a minimal-sized tree that spans *S*.



Example: *S*={ 001, 101, 110}

- NP-hard but has many heuristics (Lin & Ni '93)
- $O(t^2)$ is a trivial bound on the tree size

- Sequence alignment
- RECSA
- Existing results
- Improved solution
-

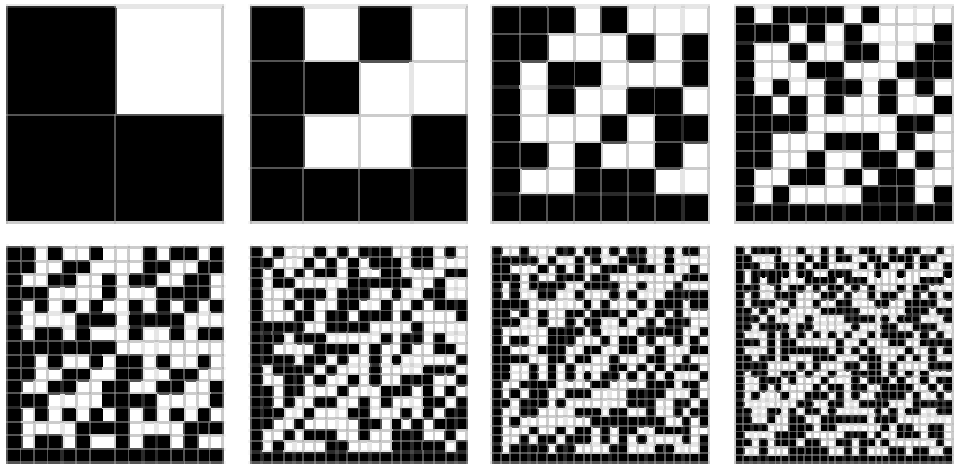
IDEA 1: Steiner tree strategy

Theorem (tight bound) The size of the Steiner minimal arborescence for a set S is $\theta(t^2)$.

Proof uses Hadamard codes that provide 2^{k+1} boolean vectors of dimension 2^k such that any two vector differ in at least 2^{k-1} positions.

Pick $t = 2^k$ vectors. For each of them, there is no other vector within the ball of radius $t/4$.

$\Rightarrow$ Steiner tree is of size $\theta(t \cdot t/4)$



- Sequence alignment
- RECSA
- Existing results
- Improved solution
-

IDEA 2: Straight-line programs

Use max of two sets (not just of a set and an element)

STRAIGHT-LINE PROGRAM (SLP) WITH BOOLEAN OPERATIONS:

Given a set S of t boolean vectors of $\{0,1\}^t$, generate all of them using an SLP $(\beta_1,\ldots,\beta_N)$ containing instructions of the following types:

1. $\beta_i := (0\ldots0,1,0\ldots0)$ (elementary vector)
2. $\beta_i := \beta_j \vee \beta_l$, with $j,l<i$ (disjunction)

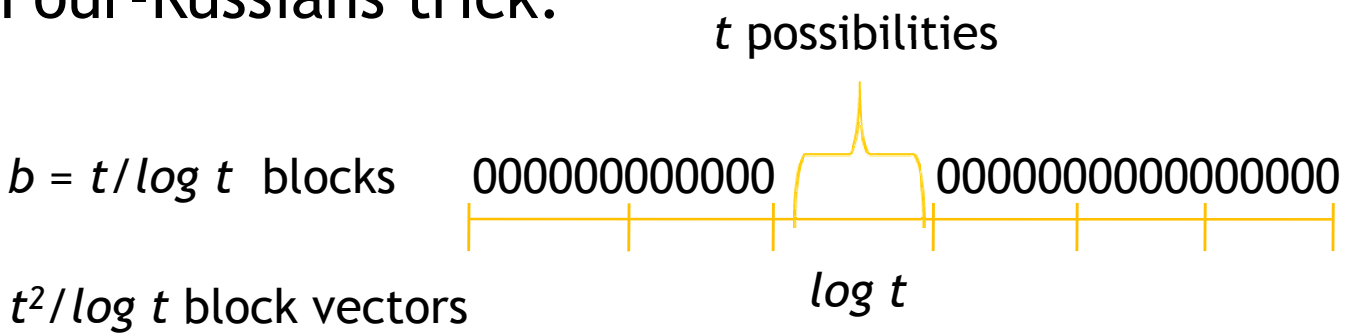
- $O(t^2)$ is a trivial bound on the SLP size

- Sequence alignment
- RECSA
- Existing results
- Improved solution
-

IDEA 2: Straight-line programs

Theorem (upper bound). An SLP for S can be constructed, with size not greater than $2t^2/\log t$ and construction time $O(t^2/\log t)$.

Four-Russians trick:



Each vector in S is included in the SLP by b-1 disjunctions of block vectors.

- Sequence alignment
- RECSA
- Existing results
- Improved solution
-

IDEA 2: Straight-line programs

Theorem (lower bound). An SLP for S has size $\Omega(t^2/\log t)$ in the worst case.

Proof by theoretic-information argument: Nb of different SLPs should be larger than the nb of different families S

Concluding remarks

- by plugging the SLP solution to maxima computation problem into the Chang et al's algorithm, we improve the $O(n^2 \cdot t \cdot |\delta|)=O(n^2 \cdot t^3)$ time bound for RECSA to $O(n^2 \cdot t \cdot t^2 / log\ t)= O(n^2 \cdot t^3 / log\ t)$

Concluding remarks

- by plugging the SLP solution to maxima computation problem into the Chang et al's algorithm, we improve the $O(n^2 \cdot t \cdot |\delta|)=O(n^2 \cdot t^3)$ time bound for RECSA to $O(n^2 \cdot t \cdot t^2 / log\ t)= O(n^2 \cdot t^3 / log\ t)$
- however, this bound is only interesting for dense NFAs, when $|\delta|$ is asymptotically larger than $O(t^2 / log\ t)$

Concluding remarks

- by plugging the SLP solution to maxima computation problem into the Chang et al's algorithm, we improve the $O(n^2 \cdot t \cdot |\delta|)=O(n^2 \cdot t^3)$ time bound for RECSA to $O(n^2 \cdot t \cdot t^2 / log\ t)= O(n^2 \cdot t^3 / log\ t)$
- however, this bound is only interesting for dense NFAs, when $|\delta|$ is asymptotically larger than $O(t^2 / log\ t)$
- ε-free NFA obtained from a regular expression of size r can be made of size $O(r \cdot log^2\ r)$ [Hromkoviec et al 01, Geffert 03, Schnitger 06]

Concluding remarks

- by plugging the SLP solution to maxima computation problem into the Chang et al's algorithm, we improve the $O(n^2 \cdot t \cdot |\delta|)=O(n^2 \cdot t^3)$ time bound for RECSA to $O(n^2 \cdot t \cdot t^2 / \log t)= O(n^2 \cdot t^3 / \log t)$
- however, this bound is only interesting for dense NFAs, when $|\delta|$ is asymptotically larger than $O(t^2 / \log t)$
- ε-free NFA obtained from a regular expression of size r can be made of size $O(r \cdot \log^2 r)$ [Hromkoviec et al 01, Geffert 03, Schnitger 06]
- $\Rightarrow$ for a regular expression of size r, RECSA can be solved in time $O(n^2 \cdot r^2 \cdot \log^2 r)$

- Sequence alignment
- RECSA
- Existing results
- Improved solution
- Concluding remarks

blue - heuristic Steiner arborescence

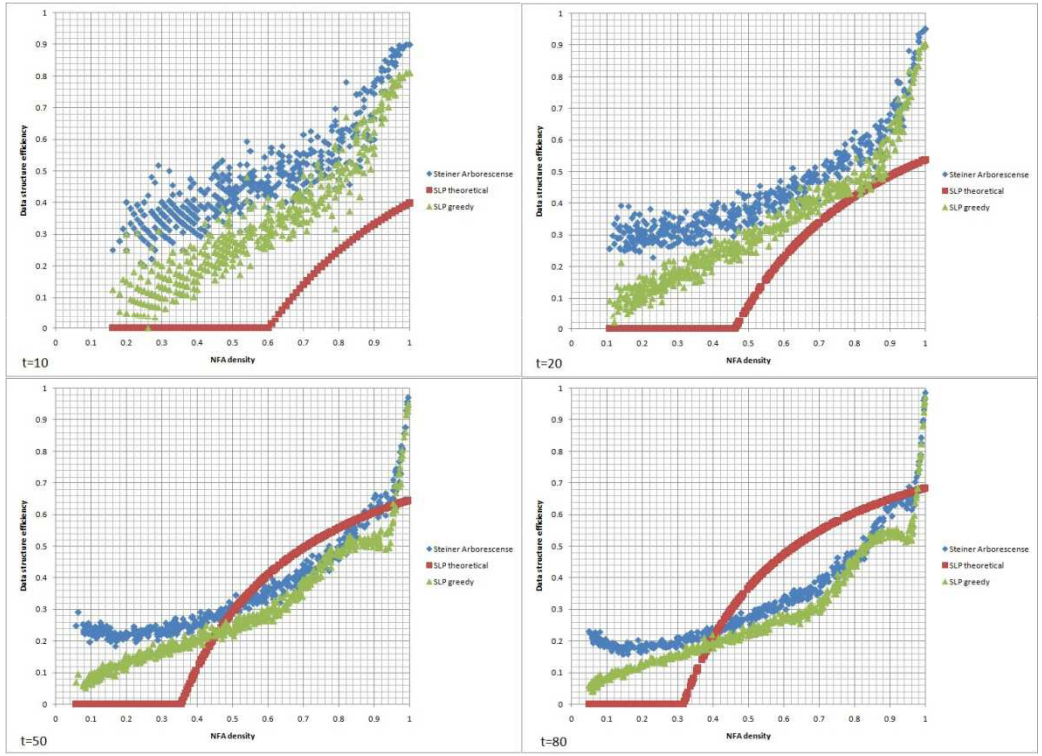
green - greedy SLP

red – Four-Russians SLP

Implementation and Experiments

`http://www.cs.bgu.ac.il/~negevcb/RL-CSA/`

Data structure efficiency = 1- (size of data structure / size of NFA transition table)



NFA density = size of NFA transition table / maximal possible no. of NFA transitions.