A Family of Approximation Algorithms for the Maximum Duo-Preservation String Mapping Problem

Bartłomiej Dudek[1] Paweł Gawrychowski[1,2]
Piotr Ostropolski-Nalewaja[1]

[1] University of Wrocław

[2] University of Haifa

July 13, 2017

Minimum Common String Partition

Input: two strings X, Y, where Y is a permutation of X.
Output: partition of X into the least number of pieces that can be rearranged (without reversing) and concatenated to obtain Y.

X : x y z a b c b x y

Y : a b b c x y z x y

Minimum Common String Partition

Input: two strings X, Y, where Y is a permutation of X.
Output: partition of X into the least number of pieces that can be rearranged (without reversing) and concatenated to obtain Y.

$$X: \quad \boxed{\texttt{x y z}}\ \boxed{\texttt{a b}}\ \boxed{\texttt{c}}\ \boxed{\texttt{b}}\ \boxed{\texttt{x y}}$$

$$Y: \quad \boxed{\texttt{a b}}\ \boxed{\texttt{b}}\ \boxed{\texttt{c}}\ \boxed{\texttt{x y z}}\ \boxed{\texttt{x y}}$$

Hardness of MCSP

Goldstein, Kolman, Zheng ['04]

MCSP is APX-hard.

Cormode, Muthukrishnan ['07]

Almost linear-time $O(\log n \cdot \log^* n)$-approximation algorithm.

Hardness of MCSP

Goldstein, Kolman, Zheng ['04]

MCSP is APX-hard.

Cormode, Muthukrishnan ['07]

Almost linear-time $O(\log n \cdot \log^* n)$-approximation algorithm.

Maximum Duo-Preservation String Mapping Problem

Complementary problem: maximize number of **duos** – consecutive letters not split apart.

$$X: \quad \boxed{\texttt{x y z}}\,\boxed{\texttt{a b}}\,\boxed{\texttt{c}}\,\boxed{\texttt{b}}\,\boxed{\texttt{x y}}$$

$$Y: \quad \boxed{\texttt{a b}}\,\boxed{\texttt{b}}\,\boxed{\texttt{c}}\,\boxed{\texttt{x y z}}\,\boxed{\texttt{x y}}$$

$$|X| = \#\mathit{duos} + \#\mathit{pieces}$$

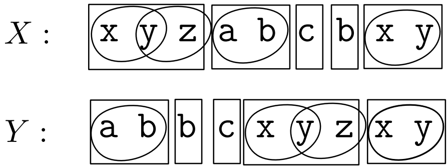
Easier? No:

Boria, Kurpisz, Leppänen, Mastrolilli ['14]:

MPSM is APX-hard.

Maximum Duo-Preservation String Mapping Problem

Complementary problem: maximize number of **duos** – consecutive letters not split apart.



Preserved duos: xy, yz, ab, xy

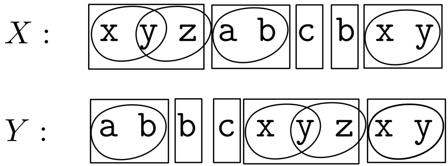
$$|X| = \#duos + \#pieces$$

Easier? No:

Boria, Kurpisz, Leppänen, Mastrolilli ['14]:
MPSM is APX-hard.

Maximum Duo-Preservation String Mapping Problem

Complementary problem: maximize number of **duos** – consecutive letters not split apart.



Preserved duos: xy, yz, ab, xy

$$|X| = \#duos + \#pieces$$

Easier? No:

Boria, Kurpisz, Leppänen, Mastrolilli ['14]:

MPSM is APX-hard.

Results for MPSM

Authors	year	ratio
Boria et al.	'14	4
Boria et al.	'16	3.5
Brubach	'16	3.25
Xu et al.	'17	2.917
DGO-N	'17	$2 + \varepsilon$

Boria, Kurpisz, Leppänen, Mastrolilli ['14]:

It is NP-hard to approximate MPSM within $1.00042 - \varepsilon$ for every $\varepsilon > 0$.

Results for MPSM

Authors	year	ratio
Boria et al.	'14	4
Boria et al.	'16	3.5
Brubach	'16	3.25
Xu et al.	'17	2.917
DGO-N	'17	$2+\varepsilon$

Boria, Kurpisz, Leppänen, Mastrolilli ['14]:

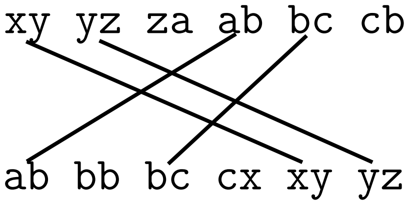
It is NP-hard to approximate MPSM within $1.00042-\varepsilon$ for every $\varepsilon>0$.

Graph representation

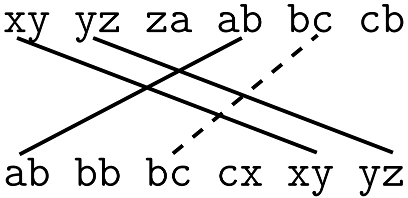
Bipartite graph with nodes – duos in both strings:

x y z a b c b

a b b c x y z

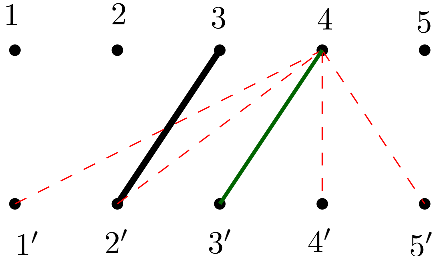


Maximum consecutive bipartite matching:



Maximum Consecutive Bipartite Matching

When we take the edge $(2', 3)$ to the matching:



Definitions

Streak:

Conflicting edges:

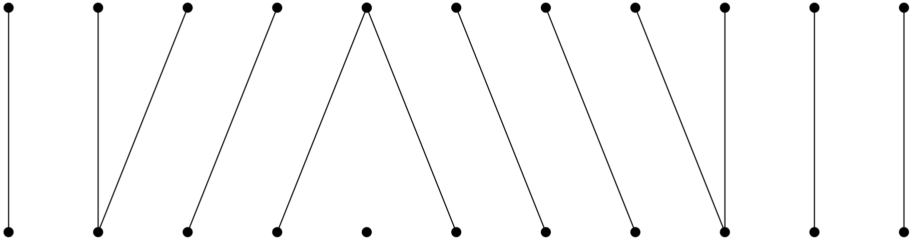
Definitions

Streak:

Conflicting edges:

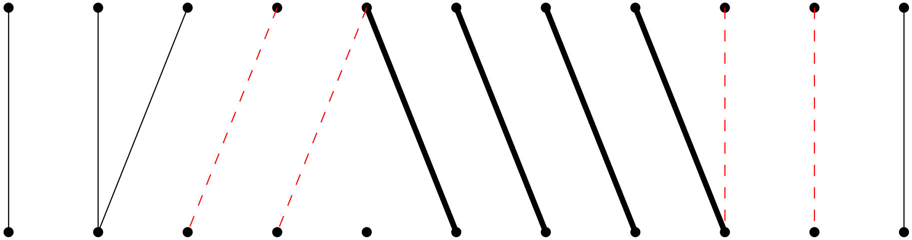
Greedy Algorithm

As long as possible take the longest possible streak from G.



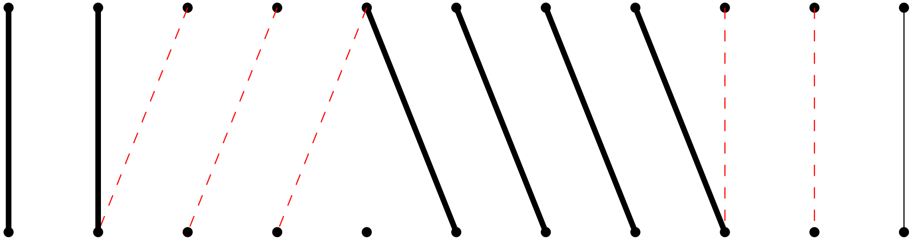
Greedy Algorithm

As long as possible take the longest possible streak from G.



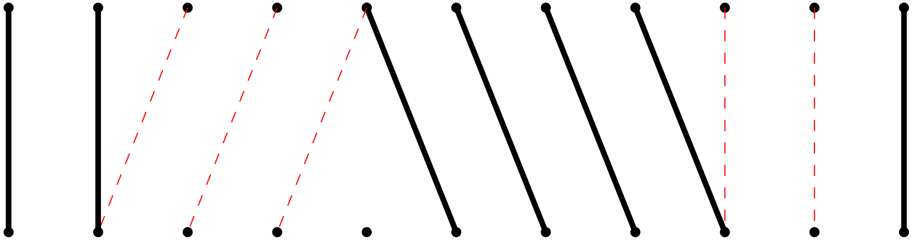
Greedy Algorithm

As long as possible take the longest possible streak from G.



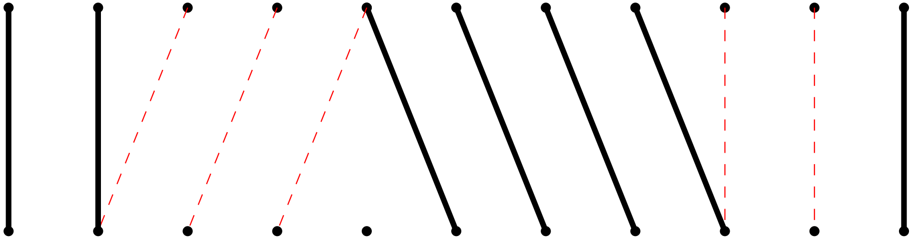
Greedy Algorithm

As long as possible take the longest possible streak from G.



Greedy Algorithm

As long as possible take the longest possible streak from G.



If we stop the algorithm when streaks contain less than k edges:

Lemma

There are $(2 + \frac{2}{k}) \cdot |\textsc{Greedy}|$ edges from optimal solution conflicting with $\textsc{Greedy}$.

Possible approaches for small k

Lemma

Streaks of length at least k: $(2 + \frac{2}{k})$-approximation.

Streaks smaller than k need another phase:

- $k = 1$ The greedy algorithm alone yields 4-approximation.
- $k = 2$ Maximum matching for the remaining edges yields 3-approximation.
- $k = 3$ Local search technique used by Boria et al. yields 2.67-approximation.

Possible approaches for small k

Lemma

Streaks of length at least k: $(2 + \frac{2}{k})$-approximation.

Streaks smaller than k need another phase:

- $k = 1$ The greedy algorithm alone yields 4-approximation.
- $k = 2$ Maximum matching for the remaining edges yields 3-approximation.
- $k = 3$ Local search technique used by Boria et al. yields 2.67-approximation.

Possible approaches for small k

Lemma

Streaks of length at least k: $(2 + \frac{2}{k})$-approximation.

Streaks smaller than k need another phase:

- $k = 1$ The greedy algorithm alone yields 4-approximation.
- $k = 2$ Maximum matching for the remaining edges yields 3-approximation.
- $k = 3$ Local search technique used by Boria et al. yields 2.67-approximation.

Possible approaches for small k

Lemma

Streaks of length at least k: $(2 + \frac{2}{k})$-approximation.

Streaks smaller than k need another phase:

- $k = 1$ The greedy algorithm alone yields 4-approximation.
- $k = 2$ Maximum matching for the remaining edges yields 3-approximation.
- $k = 3$ Local search technique used by Boria et al. yields 2.67-approximation.

$(2+\varepsilon)$-approximation

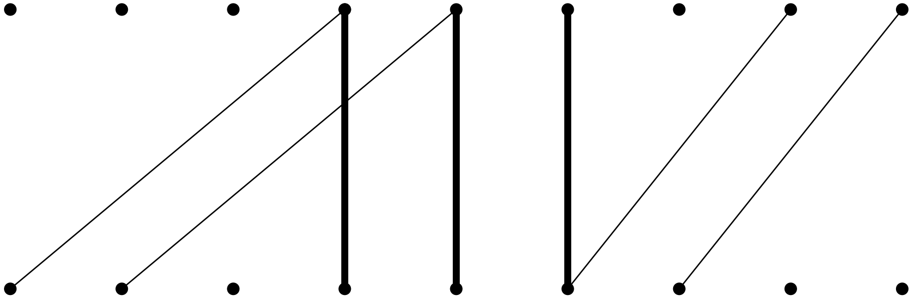
BoundedSizeImprovements(t)

- try every subset E_{add}, E_{del} of at most t edges
- if $|E_{del}| < |E_{add}|$, try $ALG \setminus E_{del} \cup E_{add}$

$(2+\varepsilon)$-approximation

BOUNDEDSIZEIMPROVEMENTS(t)

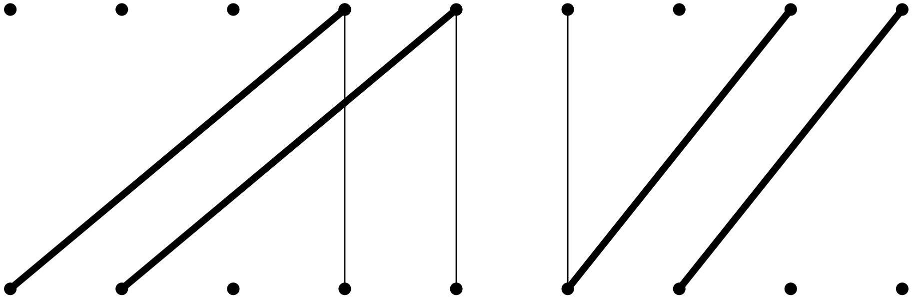
- try every subset E_{add}, E_{del} of at most t edges
- if $|E_{del}| < |E_{add}|$, try $ALG \setminus E_{del} \cup E_{add}$



$(2 + \varepsilon)$-approximation

BOUNDEDSIZEIMPROVEMENTS(t)

- try every subset E_{add}, E_{del} of at most t edges
- if $|E_{del}| < |E_{add}|$, try $ALG \setminus E_{del} \cup E_{add}$



Final algorithm

1. run GREEDY for $k = \lceil \frac{2}{\varepsilon} \rceil$
2. run BOUNDEDSIZEIMPROVEMENTS($\lceil \frac{4}{\varepsilon} \rceil + 1$).

Theorem

Combining the greedy algorithm with local improvements yields a $(2 + \varepsilon)$-approximation for MCBM in $n^{O(1/\varepsilon)}$ time, for any $\varepsilon > 0$.

Final algorithm

1. run GREEDY for $k = \lceil \frac{2}{\varepsilon} \rceil$
2. run BOUNDEDSIZEIMPROVEMENTS($\lceil \frac{4}{\varepsilon} \rceil + 1$).

Theorem

Combining the greedy algorithm with local improvements yields a $(2+\varepsilon)$-approximation for MCBM in $n^{O(1/\varepsilon)}$ time, for any $\varepsilon > 0$.

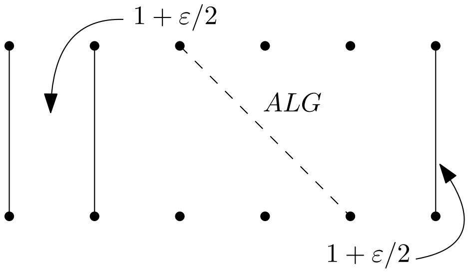
Final algorithm

1. run GREEDY for $k = \lceil \frac{2}{\varepsilon} \rceil$
2. run BOUNDEDSIZEIMPROVEMENTS($\lceil \frac{4}{\varepsilon} \rceil + 1$).

Theorem

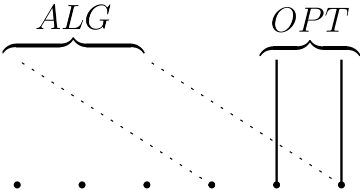
Combining the greedy algorithm with local improvements yields a $(2+\varepsilon)$-approximation for MCBM in $n^{O(1/\varepsilon)}$ time, for any $\varepsilon > 0$.

Proof: assign $2+\varepsilon$ credits to every edge from ALG



Credit distribution scheme

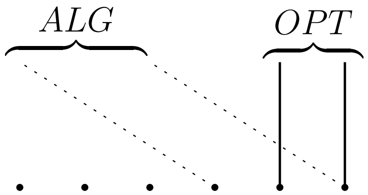
Often it is clear how to distribute the credit:



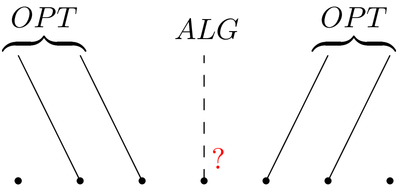
Sometimes not:

Credit distribution scheme

Often it is clear how to distribute the credit:



Sometimes not:



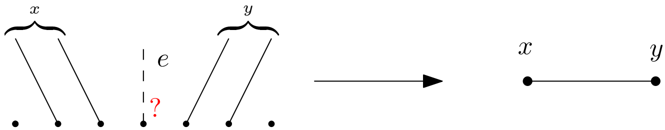
Balance of a streak

Balance: number of credits distributed to a streak minus its size.

Lemma

Balance of every streak is at least -2.

New graph with nodes – streaks of *OPT*:



Lemma

Every connected component of the new graph has overall balance at least -1.

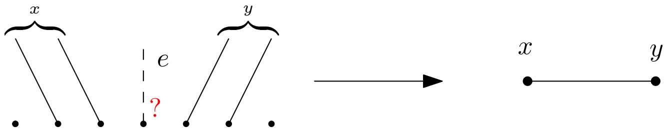
Balance of a streak

Balance: number of credits distributed to a streak minus its size.

Lemma

Balance of every streak is at least -1.

New graph with nodes – streaks of OPT:



Lemma

Every connected component of the new graph has overall balance at least -1.

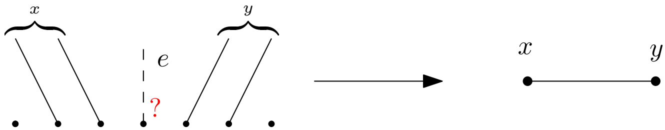
Balance of a streak

Balance: number of credits distributed to a streak minus its size.

Lemma

Balance of every streak is at least -1.

New graph with nodes – streaks of *OPT*:



Lemma

Every connected component of the new graph has overall balance at least -1.

Components with balance -1

Large components

- more than $4/\varepsilon$ credits transfered,
- unused $\varepsilon/4$ part from every credit,
- enough to cover 1 missing credit.

Small components

- not greater than t,
- conflict with less than t edeges from *ALG*,
- BOUNDEDSIZEIMPROVEMENTS(t) can improve the solution.

Components with balance -1

Large components

- more than $4/\varepsilon$ credits transfered,
- unused $\varepsilon/4$ part from every credit,
- enough to cover 1 missing credit.

Small components

- not greater than t,
- conflict with less than t edeges from ALG,
- BOUNDEDSIZEIMPROVEMENTS(t) can improve the solution.

Open problems

1. What are the actual bounds for the problem?
   - lower bound: $1.00042 - \varepsilon$
   - upper bound: $2 + \varepsilon$
2. Is k-MPSM significantly easier than MPSM?

Xu, Chen, Luo, Lin ['17]:

A $(1.4 + \varepsilon)$-approximation algorithm for the 2-MPSM.

Questions?

Open problems

1. What are the actual bounds for the problem?
   - lower bound: $1.00042 - \varepsilon$
   - upper bound: $2 + \varepsilon$
2. Is k-MPSM significantly easier than MPSM?

Xu, Chen, Luo, Lin ['17]:

A $(1.4 + \varepsilon)$-approximation algorithm for the 2-MPSM.

Questions?

Open problems

1. What are the actual bounds for the problem?
   - lower bound: $1.00042 - \varepsilon$
   - upper bound: $2 + \varepsilon$
2. Is k-MPSM significantly easier than MPSM?

Xu, Chen, Luo, Lin ['17]:

A $(1.4 + \varepsilon)$-approximation algorithm for the 2-MPSM.

Questions?