A Linear Kernel for the Complementary Maximal Strip Recovery Problem

Binhai Zhu

Computer Science Department

Montana State University

Bozeman, MT, USA

*Joint work with Haitao Jiang

Background

- For intractable problems, approximation algorithms and parameterized algorithms are the two dominant methods (which will generate results with performance guarantee). Heuristic methods (like evolutionary computation) are beyond this talk.
- In computational biology and bioinformatics, due to the inaccuracy and errors in the datasets, sometimes even a 1.5-factor approximation is useless to the biologists.
- So parameterized (or FPT) algorithms become a natural choice for many problems.
- Of course, not all problems admit FPT algorithms.

Background

An FPT algorithm for a decision problem with optimal solution value (parameter) k runs in $O(f(k)n^c)$ or $O^*(f(k))$ time, where $f(-)$ is any function only on k, c is some constant not related to k, and n is the input size.

Background

An FPT algorithm for a decision problem with optimal solution value (parameter) k runs in $O(f(k)n^c)$ or $O^*(f(k))$ time, where $f(-)$ is any function only on k, c is some constant not related to k, and n is the input size.

- For example, with Vertex Cover, instead of computing the minimum-size subset of vertices which covers all the edges, we ask “Can the edges in the input graph be covered by k vertices?”

Background

- Kernelization is a standard method (arguably the most fundamental one) in parameterized computation. Intuitively, it is data reduction. So once we have a (small) kernel for a problem, besides solving it exactly by brute-force, we can try to handle the problem with Integer Linear Programming and/or Branch-and-Bound, etc.

Kernel (formal definition)

Kernelization is a polynomial time algorithm which transforms a problem instance (I,k) into (I',k') such that:

(1) (I,k) is a yes-instance iff (I',k') is a yes-instance;

(2) $k' \leq k$; and

(3) $|I'| \leq f(k)$ for some function $f(-)$.

(I',k) or I' is usually called the kernel for the problem.

Kernel (more information)

It is well known that a problem admits an FPT algorithm iff it has a kernel.

All these info can be found in standard textbooks on FPT algorithms; e.g., Downey and Fellows (1999), Flum and Grohe (2006), and Niedermeier (2006).

Weak Kernel

While kernelization is really data reduction, weak kernel is about “search space” reduction, i.e., we are dealing with search problems.

Weak Kernel

While kernelization is really data reduction, weak kernel is about “search space” reduction, i.e., we are dealing with search problems.

There are 2 kinds of weak kernels (direct and indirect, depending on the properties of the search algorithms), the one we are talking about in this work is direct.

Problem: CMSR

- Given two comparative maps, with gene markers, we want to identify noise and redundant markers.
- In 2007, David Sankoff (U of Ottawa) first formalized this as an algorithmic problem.

Problem: CMSR

Example.

G_1=<1,2,3,4,5,6,7,8,9,10,11,12>

G_2=<-8,-5,-7,-6,4,1,3,2,-12,-11,-10,9>

Given two comparative maps, with gene markers, we want to identify noise and redundant markers.

Problem: CMSR

Example.

G_1=<1,2,3,4,5,6,7,8,9,10,11,12>

G_2=<-8,-5,-7,-6,4,1,3,2,-12,-11,-10,9>

G'_1=<1,3,6,7,8,10,11,12>

G'_2=<-8,-7,-6,1,3,-12,-11,-10>

This can be done by first finding syntenic blocks (strips) with maximum total length.

Problem: CMSR

Example.

G_1=<1,2,3,4,5,6,7,8,9,10,11,12>

G_2=<-8,-5,-7,-6,4,1,3,2,-12,-11,-10,9>

G'_1=<1,3,6,7,8,10,11,12>, 3 syntenic blocks

G'_2=<-8,-7,-6,1,3,-12,-11,-10>

G_1=<1,2,3,4,5,6,7,8,9,10,11,12>, redundant

G_2=<-8,-5,-7,-6,4,1,3,2,-12,-11,-10,9>

Definitions

- A strip (syntenic block) is a string of distinct markers that appear in two or more maps, either directly or in reversed and negated form.

  Example. 6,7,8 in G'_1; -8,-7,-6 in G'_2.

Definitions

- A strip (syntenic block) is a string of distinct markers that appear in two or more maps, either directly or in reversed and negated form.

  Example. 6,7,8 in G'_1; -8,-7,-6 in G'_2.

- MSR (Maximal Strip Recovery): Given two maps G and H, find two subsequences G' and H' of G and H, such that the total length of disjoint strips in G' and H' is maximized.

Definitions

- MSR (Maximal Strip Recovery): Given two maps G and H, find two subsequences G' and H' of G and H, such that the total length of disjoint strips in G' and H' is maximized.
- We generalize the problem to allow *d* maps, i.e., MSR-*d*.
- So MSR = MSR-2 following our generalization.
- CMSR is simply the complement of MSR, i.e., one tries to delete the minimum number of redundant markers to have a feasible solution.

Status of the Problem

- Before we started the research in 2008, the only known practical method is a heuristic based on Maximum Clique (Maximum Independent Set), by the Sankoff group.
- In 2008, we obtained a factor-4 approximation for MSR-2 (in fact, factor-2d approximation for MSR-d).
- In 2008, we showed that several close variants of MSR-2 are NP-complete; those include MSR-3, MSR-WT (i.e, when markers carry weights and the sum of weights of the markers in the strips is between w_1 and w_2), and MSR-DU (when duplicated markers are allowed).
- Both MSR and CMSR were shown to be NP-complete (Wang and Zhu, TAMC'09). MSR is APX-hard (Fertin et al.; Jiang, ISAAC'09). CMSR is also APX-hard (Jiang, FAW'10).

Status of the Problem

Approximations for CMSR:

- Factor-3 approximation: Jiang, Li, Lin, Wang and Zhu, "Exact and Approximation Algorithms for the Complementary Maximal Strip Recovery Problem", Journal of Combinatorial Optimization, 23(4):493-506, May, 2012.
- Factor-2.33 approximation: Li, Goebel, Wang and Lin, "An improved approximation algorithm for the complementary maximal strip recovery problem", Proc. FAW-AAIM'11, LNCS 6681, pp. 46-57, 2011.
- Factor-(1.5+d) approximation (for $d \geq 2$): Bulteau, Fertin, Jiang and Rusu, "Tractability and approximability of maximal strip recovery", Proc. CPM'11, LNCS 6661, pp. 336-349, 2011.

Status of the Problem

FPT Algorithms for CMSR:

- $O^*(3^k)$: Jiang, Li, Lin, Wang and Zhu, "Exact and Approximation Algorithms for the Complementary Maximal Strip Recovery Problem", Journal of Combinatorial Optimization, 23(4):493-506, May, 2012.
- $O^*(2.36^k)$: Bulteau, Fertin, Jiang and Rusu, "Tractability and approximability of maximal strip recovery", Proc. CPM'11, LNCS 6661, pp. 336-349, 2011.

Both algorithms were obtained using bounded search tree, another common technique for parameterized problems.

It is unknown whether the problem has a polynomial kernel before this work.

Our Results

(1) Let k be the minimum number of markers deleted in the optimal solution. We show that CMSR has a parameterized search space (or weak kernel) of size 18k, which is tight (for the algorithm).

Here a weak kernel for CMSR with input G_1, G_2 is a set of letters (markers) S such that one can delete k letters from it to have an optimal solution; moreover, |S| is a function of k (in our case, $|S| \leq 18k$).

Our Results

(1) Let k be the minimum number of markers deleted in the optimal solution. We show that CMSR has a parameterized search space (or weak kernel) of size 18k, which is tight (for the algorithm).

(2) This can be further transformed into a linear 84k (traditional) kernel for CMSR; combined with Bulteau et al.'s result, can yield the best FPT algorithm to this date --- $O(2.36^k k^2+n^2)$.

Technical Details

Weak Kernel for CMSR.

Lemma 1. Given input maps G_1 and G_2, if xyzw (or –w-z-y-x) appears in both G_1 and G_2 (as 4-substring), then there is an optimal solution for MSR which has xyzw (or –w-z-y-x) as a strip.

Example:

G_1=-w-zxabcy, G_2=zabcw-y-x

Technical Details

Weak Kernel for CMSR.

Lemma 1. Given input maps G_1 and G_2, if xyzw (or –w-z-y-x) appears in both G_1 and G_2 (as 4-substring), then there is an optimal solution for MSR which has xyzw (or –w-z-y-x) as a strip.

Example:

G_1=-w-zxabcy, G_2=zabcw-y-x

We can't keep abc as a strip.

Technical Details

Weak Kernel for CMSR.

Lemma 1. Given input maps G_1 and G_2, if xyzw (or –w-z-y-x) appears in both G_1 and G_2 (as 4-substring), then there is an optimal solution for MSR which has xyzw (or –w-z-y-x) as a strip.

Initial Weak Kernelization (incomplete):

1. Without deleting any gene marker, identify a set of blocks (maximal common substrings) in G_1 and G_2, with length at least 4.

2. For each block identified, change it to a new letter in $\sum_1$, with $\sum_1 \cap \sum = \emptyset$ ($\sum$ is the set of input markers). Let the resulting sequences be G'_1 and G'_2.

Technical Details

Weak Kernel for CMSR.

Initial Weak Kernelization (incomplete):

1. Without deleting any gene marker, identify a set of blocks in G_1 and G_2, with length at least 4.

2. For each block identified, change it to a new letter in $\sum_1$, with $\sum_1 \cap \sum = \emptyset$ ($\sum$ is the set of input markers). Let the resulting sequences be G'_1 and G'_2.

Why it is incomplete?

Technical Details

Weak Kernel for CMSR.

Initial Weak Kernelization (incomplete):

1. Without deleting any gene marker, identify a set of blocks in G_1 and G_2, with length at least 4.

2. For each block identified, change it to a new letter in $\sum_1$, with $\sum_1 \cap \sum = \emptyset$ ($\sum$ is the set of input markers). Let the resulting sequences be G'_1 and G'_2.

What about length-2 and length-3 blocks, e.g.,

Xabc def ghi

Xghi def abc

Technical Details

Weak Kernel for CMSR.

Initial Weak Kernelization (first fix-up):

1. Without deleting any gene marker, identify a set of blocks in G_1 and G_2, with length at least 4.

And identify length-2 and length-3 blocks which could appear in some optimal solution accordingly, using local rules.

2. For each block identified, change it to a new letter in $\sum_1$, with $\sum_1 \cap \sum = \emptyset$ ($\sum$ is the set of input markers). Let the resulting sequences be G'_1 and G'_2.

What about length-2 and length-3 blocks, e.g.,

Xabc def ghi

Xghi def abc

Technical Details

Weak Kernel for CMSR.

Initial Weak Kernelization (first fix-up):

1. Without deleting any gene marker, identify a set of blocks in G_1 and G_2, with length at least 4.

And identify length-2 and length-3 blocks which could appear in some optimal solution accordingly, using local rules.

2. For each block identified, change it to a new letter in $\sum_1$, with $\sum_1 \cap \sum = \emptyset$ ($\sum$ is the set of input markers). Let the resulting sequences be G'_1 and G'_2.

What about length-2 and length-3 blocks, e.g.,

Xabc def ghi

Xghi def abc

E.g., how about a length-3 block with at most 1 isolated neighbor?

Technical Details

Initial Weak Kernelization (first fix-up):

1. Without deleting any gene marker, identify a set of blocks in G_1 and G_2, with length at least 4.

And identify length-2 and length-3 blocks which could appear in some optimal solution accordingly, using local rules.

2. For each block identified, change it to a new letter in $\sum_1$, with $\sum_1 \cap \sum = \emptyset$ ($\sum$ is the set of input markers). Let the resulting sequences be G'_1 and G'_2.

What about length-2 and length-3 blocks, e.g.,

Xabc def ghi

Xghi def abc

E.g., how about a length-3 block with at most 1 isolated neighbor?

$G1 = xP_1QP_2y\ a_1b_1\ a_2b_2\ a_3P_3b_3\ a_4P_4b_4\ zw$ Q has length-3

$G2 = zP_3QP_4w\ a_4b_4\ a_3b_3\ a_1P_1b_1\ a_2P_2b_2\ xy$ P_i has length-2

Technical Details

Initial Weak Kernelization (first fix-up):

1. Without deleting any gene marker, identify a set of blocks in G_1 and G_2, with length at least 4.

And identify length-2 and length-3 blocks which could appear in some optimal solution accordingly, using local rules.

2. For each block identified, change it to a new letter in $\sum_1$, with $\sum_1 \cap \sum = \emptyset$ ($\sum$ is the set of input markers). Let the resulting sequences be G'_1 and G'_2.

What about length-2 and length-3 blocks, e.g.,

Xabc def ghi

Xghi def abc

E.g., how about a length-3 block with at most 1 isolated neighbor?

$G1 = x P_1 Q P_2 y\ a_1 b_1\ a_2 b_2\ a_3 P_3 b_3\ a_4 P_4 b_4\ zw$ Opt deletes Q

$G2 = z P_3 Q P_4 w\ a_4 b_4\ a_3 b_3\ a_1 P_1 b_1\ a_2 P_2 b_2\ xy$ and P_is

Technical Details

Weak Kernel for CMSR.

Initial Weak Kernelization (first fix-up):

1. Without deleting any gene marker, identify a set of blocks in G_1 and G_2, with length at least 4.

And identify length-2 and length-3 blocks which could appear in some optimal solution accordingly, using local rules.

2. For each block identified, change it to a new letter in $\sum_1$, with $\sum_1 \cap \sum = \emptyset$ ($\sum$ is the set of input markers). Let the resulting sequences be G'_1 and G'_2.

What about length-2 and length-3 blocks?

Well, except for 1, we found counter-examples for any local rules we could think of!

Technical Details

Weak Kernel for CMSR.

Final Weak Kernelization:

1. Without deleting any gene marker, identify a set of blocks in G_1 and G_2, with lengths ≥4, 3, 2, 1. Call each maximal continuous blocks, each of length ≥ 2, a super-block. Let the set of super-blocks in G_i be V_i.

2. (2.1) for each block of length ≥ 4, change it to a new letter in $\sum_1$, with $\sum_1 \cap \sum = \emptyset$ ($\sum$ is the set of input markers).

(2.2) for super-blocks $s_1 \epsilon V_1$, $s_2 \epsilon V_2$ which contain at least 2 pairs of common (length-2 or length-3) blocks, identify the leftmost and rightmost such blocks, e.g., P_i, P_j in s_1, P_l, P_r in s_2. Change each block between (and inclusive of) them into a new letter in $\sum_1$.

Example: s_1 = ab cd ef gh, s_2 = ab xy zw gh, each of these 6 length-2 blocks should be kept in some optimal solution.

Technical Details

Weak Kernel for CMSR.

Final Weak Kernelization (Step 2):

2. (2.1) for each block of length ≥ 4, change it to a new letter in $\sum_1$, with $\sum_1 \cap \sum = \emptyset$ ($\sum$ is the set of input markers).

(2.2) for super-blocks $s_1 \epsilon V_1$, $s_2 \epsilon V_2$ which contain at least 2 pairs of common (length-2 or length-3) blocks, identify the leftmost and rightmost such blocks, e.g., P_i,P_j in s_1, P_l,P_r in s_2. Change each block between (and inclusive of) them into a new letter in $\sum_1$.

(2.3) for any super-block containing at least two length-3 blocks, identify the leftmost and rightmost ones, say P_s, P_t. Change each block between and inclusive of P_s, P_t into a new letter in $\sum_1$.

Example: s = xy abc de fgh ij klm zw, then abc, de, fgh, ij, klm should be kept in some optimal solution.

Technical Details

Weak Kernel for CMSR.

Final Weak Kernelization (Step 2):

2. (2.1) for each block of length ≥ 4, change it to a new letter in $\sum_1$, with $\sum_1 \cap \sum = \emptyset$ ($\sum$ is the set of input markers).

(2.2) for super-blocks $s_1 \epsilon V_1$, $s_2 \epsilon V_2$ which contain at least 2 pairs of common (length-2 or length-3) blocks, identify the leftmost and rightmost such blocks, e.g., P_i, P_j in s_1, P_l, P_r in s_2. Change each block between (and inclusive of) them into a new letter in $\sum_1$.

(2.3) for any super-block containing at least two length-3 blocks, identify the leftmost and rightmost ones, say P_s, P_t. Change each block between and inclusive of P_s, P_t into a new letter in $\sum_1$.

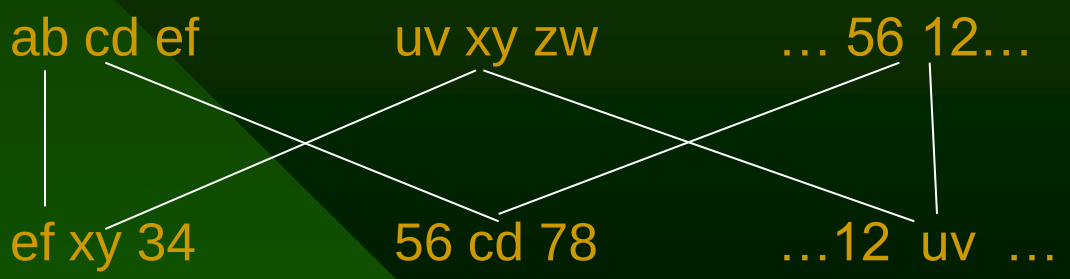
(2.4) construct a bipartite graph $G=(V_1,V_2,E)$, where an edge $(v_1,v_2) \epsilon E$ iff they share a common block (of length 2 or 3) not yet put in $\sum_1$. For any cycle in G, identify the length-2 (and 3) blocks involved with the cycle and change each such block to a letter in $\sum_1$.

Technical Details

(2.3) for any super-block containing at least two length-3 blocks, identify the leftmost and rightmost ones, say P_s, P_t. Change each block between and inclusive of P_s, P_t into a new letter in $\sum_1$.

(2.4) construct a bipartite graph $G=(V_1,V_2,E)$, where an edge $(v_1,v_2)\epsilon E$ iff they share a common block (of length 2 or 3) not yet put in $\sum_1$. For any cycle in G, identify the length-2 (and 3) blocks involved with the cycle and change each such block to a letter in $\sum_1$.

Example:



Then, cd, xy, 12, ef, uv, 56 (six length-2 blocks) should be kept in some optimal solution.

Technical Details

(2.3) for any super-block containing at least two length-3 blocks, identify the leftmost and rightmost ones, say P_s, P_t. Change each block between and inclusive of P_s, P_t into a new letter in $\sum_1$.

(2.4) construct a bipartite graph $G=(V_1,V_2,E)$, where an edge $(v_1,v_2) \epsilon E$ iff they share a common block (of length 2 or 3) not yet put in $\sum_1$. For any cycle in G, identify the length-2 (and 3) blocks involved with the cycle and change each such block to a letter in $\sum_1$.

(2.5) within any super-block, for all blocks between two letters in $\sum_1$, change each of them into a new letter in $\sum_1$.

$\sum_1$ $\sum_1$

Example: s = abcd xy zw 1234, yellow block is a letter in $\sum_1$.

then, xy, zw should be kept in some optimal solution.

Technical Details

(2.3) for any super-block containing at least two length-3 blocks, identify the leftmost and rightmost ones, say P_s, P_t. Change each block between and inclusive of P_s, P_t into a new letter in $\sum_1$.

(2.4) construct a bipartite graph $G=(V_1,V_2,E)$, where an edge $(v_1,v_2) \epsilon E$ iff they share a common block (of length 2 or 3) not yet put in $\sum_1$. For any cycle in G, identify the length-2 (and 3) blocks involved with the cycle and change each such block to a letter in $\sum_1$.

(2.5) within any super-block, for all blocks between two letters in $\sum_1$, change each of them into a new letter in $\sum_1$.

3. Return $S \leftarrow \sum$ as the parameterized search space (weak kernel).

Remember that in rules 2.1-2.5, once we change some block into a new letter in $\sum_1$, all its previous letters in $\sum$ must be deleted from $\sum$.

Technical Details

Weak Kernel for CMSR.

We need to show that $|S| \leq 18k$, which is done through an inverse amortized analysis.

The setting of some weights in the proof is helped with a matching lower bound.

Technical Details

Weak Kernel for CMSR.

We need to show that $|S| \leq 18k$, which is done through an inverse amortized analysis.

The setting of some weights in the proof is helped with a matching lower bound.

Matching lower bound:

G_1= abc de fxg hij kl mn opq Opt deletes x.

G_2= abc fg de hij mxn kl opq

Technical Details

Weak Kernel for CMSR.

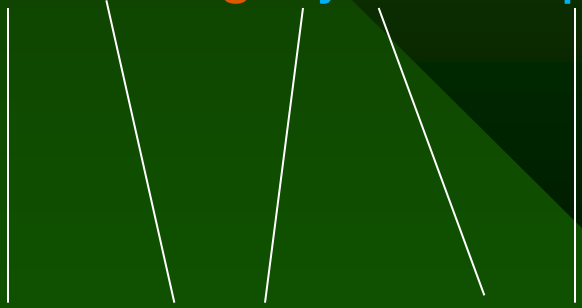
We need to show that $|S| \leq 18k$, which is done through an inverse amortized analysis.

The setting of some weights in the proof is helped with a matching lower bound.

Matching lower bound:

G_1= abc de fxg hij kl mn opq

Opt deletes x.



G_2= abc fg de hij mxn kl opq

Technical Details

Theorem: CMSR has a weak kernel of size 18k.

Corollary. With an easy counting method, CMSR admits a kernel of size 84k.

---Note that in the counting each letter in $\sum_1$ will be counted as of length-4, whereas in the original input such a block could be of length bigger than 4, say 25.

---A continuous sequence of $\sum_1$ letters can be compressed by one new letter in $\sum_1$.

Technical Details

Theorem: CMSR has a weak kernel of size 18k.

Corollary. With an easy counting method, CMSR admits a kernel of size 84k.

The 18k weak kernel bound is tight with respect to our method --- but this does not exclude the possibility of improving it using a more involved method.

Summary

Theorem: CMSR has a weak kernel of size 18k.

Corollary. With an easy counting method, CMSR admits a kernel of size 84k.

Corollary. Combined with the bounded search tree method, CMSR can be solved in $O(2.36^k k^2+n^2)$ *time*.

Direct vs Indirect weak kernel

- It seems that for problems admitting direct weak kernels, bounded search trees work and small kernels exist, at least for CMSR (18k) and Min co-Path Set (5k), moreover; they can be converted into traditional kernels (84k,5k---COCOA'12).
- For a class of problems admitting indirect weak kernels (sorting by reversals, sorting by DCJ, etc), small weak kernels exist (4k,2k) but no efficient bounded search tree exists and no polynomial kernel is known.

Open Problems

1. *Improvement of the $O^*(2.36^k)$ FPT algorithm? For practical datasets, n could be roughly 700 and k could be 120.*
2. *New algorithmic applications of weak kernels?*
3. Identify problems in NP which could not have small weak kernels.