Electronic Colloquium on Computational Complexity, Revision 2 of Report No. 5 (2010)

Weak Kernels

Haitao Jiang[*] Binhai Zhu[†]

May 4, 2010

Abstract

In this paper, we formalize a folklore concept and formally define *weak kernels* for fixed-parameter computation. We show that a problem has a (traditional) kernel then it also has a weak kernel. It is unknown yet whether the converse is always true. On the other hand, for a problem in NP, if it has a weak kernel then it admits an FPT algorithm (hence a kernel). We show a few applications of weak kernels, for which a (traditional) kernelization seems hard to apply. Among them, we present the first FPT algorithm for the famous Sorting by Minimum Unsigned Reversals problem.

[*] Department of Computer Science, Montana State University, Bozeman, MT 59717, USA. Email: htjiang@cs.montana.edu.

[†] Corresponding author. Department of Computer Science, Montana State University, Bozeman, MT 59717, USA. Email: bhz@cs.montana.edu.

ISSN 1433-8092

1 Introduction

In the last four decades, we have seen the huge advance of NP-completeness [12, 26, 18]. Nowadays, NP-complete problems appear in almost all the areas which involves combinatorial optimization, for example in computational biology and bioinformatics. As from the beginning a lot of people tended to believe $P \neq NP$ (at least it seems to be hard to prove or disprove it), people immediately started to investigate different ways to handle NP-hard problems. Up to today, the two most popular ways to handle NP-hard problems, among researchers in algorithm design, are approximation algorithms and exact (or FPT) algorithms, which were started with the seminal works of Johnson [24] and Tarjan and Trojanowski [30] respectively. (Using heuristic methods to hand NP-hard problems, like evolutionary computation, is beyond this paper.)

In some areas like computational biology and bioinformatics, the data usually contain errors. On top of this, if we design a factor-2 approximation to handle these data, whatever result we got is not appealing to biologists. So, to make approximation algorithms useful for these applications, the approximation factors must be very close to one. Then, naturally, FPT algorithm pops up as a natural alternative for handling these problems. The three applications we will discuss in this paper all originate from computational biology.

On the other hand, the theory of fixed-parameter computation has been developed rigorously in the last two decades. The first textbook was published in 1999 by Downey and Fellows [14] and another couple were published in the last several years. Interested readers are referred to [19] for further details and references.

In designing FPT algorithms, kernelization is one of the most fundamental techniques. Loosely speaking, kernelization is really *data reduction*; i.e., with kernelization one reduces the problem instance size (kernel size) to a level so small that one could even apply a brute-force method. Sometimes, even if the kernel size is slightly bigger (say 2^k) so that a brute-force method is inappropriate, one can still make use of it with integer linear programming or branch-and-bound to obtain almost optimal solutions in a reasonable amount of time [19].

In this paper, we formalize a folklore method and formally define *weak kernels* and weak kernelization. Again, loosely speaking, when viewing an NP-hard optimization problem as a searching or decision problem (like for Vertex Cover, we are really searching for a set of k vertices, among n input vertices, so that deleting the k vertices leaves the resulting graph edge-less), weak kernelization is really about *search space reduction*. Certainly, if a problem has a kernel then of course it has a weak kernel (which we will prove formally). But whether the converse is true is unknown yet. We show formally that for problems in NP the converse is in fact true.

The purpose for defining the weak kernels concept, on the other hand, is more on helping us design FPT algorithms more easily. Here, we show three applications, all known to be NP-complete, for which we compute the corresponding weak kernels efficiently (hence design efficient FPT algorithms). Among the three problems, Sorting with Minimum Unsigned Reversals is a famous problem in computational biology and we do not know of any non-trivial kernelization or FPT algorithm for it. We show that Sorting with Minimum Unsigned Reversals has a weak kernel of size $4k$, hence an FPT algorithm running in $O(2^{4k}n + n\log n)$ time (and with a more detailed analysis, in $O(2^{2k}n + n\log n)$ time).

This paper is organized as follows. In Section 2, we define weak kernels formally and prove its relation with the (traditional) kernels. In Section 3, we show three applications of weak kernels. In Section 4, we conclude the paper with several open problems.

2 Kernels vs Weak Kernels

2.1 Preliminaries

Basically, a fixed-parameter tractable (FPT) algorithm for a decision problem Π with solution value k is an algorithm which solves the problem in $O(f(k)n^c)$ time, where f is any function only on k, n is the input size and c is some fixed constant not related to k. FPT also stands for the set of problems which admit such an algorithm.

Kernelization is a polynomial time transformation that transforms a problem instance (I, k) to another instance (I', k') such that (1) (I, k) is a yes-instance iff (I', k') is also a yes-instance; (2) $k' \leq k$; and (3) $|I'| \leq f(k)$ for some function $f(-)$. (I', k) is typically called a *kernel* of the problem, with size $|I'|$. It is easy to see that if a problem has a kernel then it is in FPT; moreover, every problem in FPT has a kernel. From this, kernelization is a way to perform data-reduction with performance guarantee, "a humble strategy for coping with hard problems, almost universally employed" [15]. More fundamental details on FPT algorithms can be found in [14].

Recently, Bodlaender *et al.* conducted a seminal work by showing that a class of important FPT problems cannot have polynomial (e.g., $O(k^2)$ size) kernels unless the polynomial hierarchy collapses to the third level (i.e., $PH = \Sigma_p^3$) [5]. One such problem is called k-LEAF OUT-BRANCHING (i.e., finding a rooted oriented spanning tree with at least k leaves in an input digraph $\mathcal{D}$) [17]. On the other hand, Fernau *et al.* showed that the ROOTED k-LEAF OUT-BRANCHING has a polynomial kernel, hence implying that k-LEAF OUT-BRANCHING has a Turing kernelization [17].

2.2 Weak Kernels

As illustrated in the introduction, we view weak kernelization (weak kernels) as a way to reduce search space. Given a decision problem Π, let $\Pi(I)$ be an instance of Π, and let a solution of size k can be searched from a space $S(I)$ which is a component of $\Pi(I)$ or can be constructed from some components of $\Pi(I)$. So we denote the resulting search problem as $(\Pi(I), S(I), k)$. For example, the corresponding search problem for Vertex Cover is $(G = (V, E), V, k)$. Similarly, for k-LEAF OUT-BRANCHING, the corresponding search problem is $(\mathcal{D} = (V, E), L, k)$, where L represents the super-set of vertices each corresponding to the set of leaves belonging to some rooted oriented spanning trees of $\mathcal{D}$.

A *weak kernelization* is a polynomial time transformation which transforms a search problem instance $(\Pi(I), S(I), k)$ into $(\Pi(I'), S'(I), k)$ such that (1) $|\Pi(I')| \leq |\Pi(I)|$; (2) $|S'(I)| \leq f(k)$ for some function $f(-)$; (3) $(\Pi(I), S(I), k)$ is a yes-instance iff $(\Pi(I'), S'(I), k)$ is a yes-instance. Note that condition (1) is not important in our definition, in other words, while we have to reduce search space, we can but do not have to reduce the problem input size (e.g., we can even make $\Pi(I) = \Pi(I')$). $(\Pi(I'), S'(I), k)$ (or simply $S'(I)$) is also called the corresponding *weak kernel* for Π, with size $|S'(I)|$.

We have the following results regarding weak kernels.

Lemma 1 *If a problem Π has a kernel, then it has a weak kernel.*

Proof. Let the kernel for $\Pi(I)$ be $(\Pi(I'), k')$, with $|\Pi(I')| \leq f(k)$ for some function $f(-)$ and $k' \leq k$. In other words, in the preprocessing steps to construct the kernel, $k - k'$ items are deleted (or included in every optimal solution). We can construct a new instance $\Pi(I_1)$ by putting these $k - k'$ items and the associated data back into $\Pi(I')$. Now we can view the (searching) problem Π as $(\Pi(I_1), S(I_1), k)$, where $S(I_1)$ is the search space formed by searching k' items from $\Pi(I')$ and the $k - k'$ items which have just been put back into $\Pi(I')$ (to obtain $\Pi(I_1)$). This is clearly the corresponding weak kernel; i.e., $\Pi(I)$ can be solved by searching a solution of size k from $S(I_1)$, with $|S(I_1)| \leq |\Pi(I')| + (k - k') \leq f(k) + k$. Hence, Π has a weak kernel. □

The converse of Lemma 1 is not necessarily true (at least we cannot prove that the converse is true at this point). For instance, let an EXP-complete search problem $(\Pi'(I), S(I), k)$ have a weak kernel of size $w(k)$, say $(\Pi'(I), S'(I), k)$ with $S'(I) \leq w(k)$. As Π' is EXP-complete, checking whether a solution of size k from $S'(I)$ is a YES/NO instance for $\Pi'(I)$ probably cannot be done in polynomial time (unless the complexity hierarchy collapses at certain place). How to find such an problem Π' beyond NP is open.

On the other hand, when $\Pi \in NP$, we can show that traditional kernels and weak kernels are equivalent in theory. Of course, that does not imply their sizes are the same.

Theorem 1 *If a problem $\Pi \in NP$ has a weak kernel, then it admits an FPT algorithm.*

Proof. Let the weak kernel be $(\Pi(I), S'(I), k)$, with $|S'(I)| \leq f(k)$ for some function $f(-)$. We can enumerate all possible solutions of size k in $S'(I)$ and for each one check whether it is a YES/NO instance, which can be done in polynomial time as $\Pi \in NP$. If no YES instance is found then return NO; otherwise return the YES instance. Hence we have an FPT algorithm. □

Corollary 1 *If a problem $\Pi \in NP$ has a weak kernel, then it has a kernel.*

Proof. If Π has a weak kernel, following Theorem 1, it admits an FPT algorithm. Following the known result in FPT theory, a problem in FPT always has a kernel [14]. Hence Π has a kernel. □

Combining this with Lemma 1, we have the following theorem.

Theorem 2 *A problem $\Pi \in NP$ has a weak kernel if and only if it has a kernel.*

While the above proofs are not really difficult, they have interesting theoretical implications. For example, if one could find a problem in PSPACE which has a weak kernel but not a kernel, then NP≠PSPACE. For another instance, for a problem unlikely to have a kernel (say k-Dominating Set, which is W[2]-complete), as long as it belongs to NP, it is equally unlikely to have a weak kernel. Therefore, for problems in NP, the true merit of the above results seems to be helping us design efficient FPT algorithms via weak kernels directly. (Through a private communication with Mike Fellows, the earliest idea of using weak kernels seems to be in [1], where Bonsma, Brüggemann and Woeginger showed that the MAX LEAF problem has a weak kernel of size $3.5k$. Note that MAX LEAF is the complement of the Minimum Connected Dominating Set problem.) We show below three examples of the applications of weak kernels. For all of them, we do not know of better kernel bounds. For the famous Sorting with Minimum Unsigned Reversals, this is the first non-trivial FPT algorithm.

3 Applications

We consider three minimization problems which are all known to be NP-complete: Complement of Maximal Strip Recovery (CMSR), Minimum Co-Path Set and Sorting with Minimum Unsigned Reversals (SMUR).

3.1 CMSR

Given two genomic maps G_1 and G_2 represented by a sequence of n gene markers, a *strip* (syntenic block) is a sequence of distinct markers of length at least two which appear as subsequences in the input maps, either directly or in reversed and negated form. The problem *Maximal Strip Recovery* (MSR) is to find two subsequences $G_1^\star$ and $G_2^\star$ of G_1 and G_2, respectively, such that the total length of disjoint strips in $G_1^\star$ and $G_2^\star$ is maximized (i.e., conversely, the complement of the problem CMSR is to minimize the number of markers deleted to have a feasible solution). An example of MSR and CMSR is shown in the following figure, in which each integer represents a marker. Throughout this paper, a sequence is either denoted as a list as in Figure 1, or it can just be denoted as a string.

$$\begin{aligned}
G_1 &= \langle 1,2,3,4,5,6,7,8,9,10,11,12\rangle \\
G_2 &= \langle -9,-4,-7,-6,8,1,3,2,-12,-11,-10,-5\rangle \\
S_1 &= \langle 1,2\rangle \\
S_2 &= \langle 6,7,9\rangle \\
S_3 &= \langle 10,11,12\rangle \\
G_1^\star &= \langle 1,2,6,7,9,10,11,12\rangle \\
G_2^\star &= \langle -9,-7,-6,1,2,-12,-11,-10\rangle
\end{aligned}$$

Figure 1: An example for the problem MSR and CMSR. MSR has a solution size of eight, with three strips S_1, S_2, S_3. CMSR has a solution size of four: the deleted markers are 3,4,5 and 8.

MSR and CMSR were motivated by eliminating redundancies in genomic maps [11, 35]. Recently, both MSR and CMSR are shown to be NP-complete [33], in fact, APX-hard [6, 22, 23]. The generalization to handle more than two genomic maps can be found in [8, 33].

The computation of weak kernel for CMSR was already presented in [33]. (Of course, it was a folklore and was even called 'kernel' in [33]; likewise, MSR and CMSR was not properly distinguished in [33] even though they are complement to each other.) Due to the mis-use of terminology, missing of details and confusion, we sketch and revise the crucial parts of the method to have a clear one here, as the first application of weak kernels.

Before any marker is deleted, we can identify all maximal common substrings of length at least one (possibly in negated and reversed form, which will also be called maximal common substrings for convenience) of G_1 and G_2. We also call a length-1 maximal common substring (which is a letter) an *isolated* letter or *isolate*. Two substrings are called *neighbors* if there is no other string in between them.

Lemma 2 *Before any marker is deleted, if a length-4 maximal common substring $xyzw$ or $-w-z-y-x$ appears in both G_1 and G_2 (or, if $xyzw$ appears in G_1 and $-w-z-y-x$ appears in*

G_2, and vice versa), then there is an optimal solution for MSR which has $xyzw$ or $-w-z-y-x$ as a strip.

Proof. Wlog, we only consider the case when $xyzw$ appears in G_1 and $-w-z-y-x$ appears in G_2. The cases when $xyzw$ ($-w-z-y-x$) appears in both G_1 and G_2 are similar.

Let the length-6 substring in G_1 containing $xyzw$ be $p_1(x)xyzws_1(w)$ and let the length-6 substring in G_2 containing $-w-z-y-x$ be $p_2(w)-w-z-y-xs_2(x)$. When delete $xyzw$ from G_1 and $-w-z-y-x$ from G_2, at most two strips can be obtained: $p_1(x)s_1(w)$ and $p_2(w)s_2(x)$ (with a total length of 4). Clearly, retaining $xyzw$ and $-w-z-y-x$ as a strip can give us a solution at least as good as any optimal solution. Hence, the lemma is proven. □

The above lemma holds for maximal common substrings of length greater than 4. In fact, similar to that, we can show that a length-3 maximal common substring of G_1 and G_2 which has at most 3 isolated neighbors in G_1 and G_2 can be a strip in some optimal solution of MSR, etc.

Let Σ be the alphabet for the input maps G_1 and G_2. The above results give us a weak kernelization procedure. (We comment that the original weak kernelization in [33] is incomplete — thanks to a friend, who does not want his name exposed, for pointing that out, though the size of the weak kernel is not affected.)

1. Without deleting any gene marker in G_1 and G_2, identify a set of maximal common substrings of length at least four, a set of maximal common substrings of length three which has at most 3 isolated neighbors, and a set of maximal common substrings of length two which has at most 2 isolated neighbors, from the two sequences G_1 and G_2.

2. For each common substring identified, change it to a new letter in Σ_1, with $\Sigma_1 \cap \Sigma = \emptyset$. Let the resulting sequences be G_1', G_2'.

The correctness of this procedure follows from the fact that if a maximal common substring S of length-p in G_1 and G_2 has q isolated neighbors and $q \leq p$, then S is a strip in some optimal solution of MSR. (If not, then we could delete the q isolated neighbors of S, make S a strip and hence obtain a solution at least as good as before.) Consequently, we can focus on a special kind of solution for CMSR in which the maximum number of isolated letters are deleted.

Let Σ_1 be the set of new letters used in the weak kernelization process, with $\Sigma_1 \cap \Sigma = \emptyset$. The two lemmas for obtaining the final result are: (1) There is an optimal CMSR solution of size k for G_1 and G_2 if and only if the solution can be obtained by deleting k markers in Σ from G_1' and G_2' respectively. (2) In G_1' (resp. G_2'), there are at most $5k$ letters (markers) in Σ [33]. To see a slightly revised proof of the last lemma (due to the revised weak kernelization procedure), let k_i be number of length-i common substrings deleted in the optimal CMSR solution. Consequently we have

$$k = k_1 + 2k_2 + 3k_3.$$

The size of the weak kernel is the number of letters that can possibly be deleted, i.e., in Σ. Let S be a length-p maximal common substring to be deleted. If $p = 3$, then S has at most 4 isolated neighbors and we have 7 associated letters for S. If $p = 2$, then S has 3 or 4 isolated neighbors; and we can have 6 associated letters for S (when we have 4 isolated neighbors), 7 or 8 (three isolated neighbors and another neighbor of length-2 or length-3). Now let us consider the remaining letters which are all isolates. Let x be a marker to be deleted, let it appear in G'_1 as $\cdots axb \cdots cd \cdots$ and let it appear in G'_2 as $\cdots cxd \cdots ab \cdots$. Clearly, in this example x is associated with $\{x, a, b, c, d\}$. In other words, for each isolate, we have 5 associated letters. Putting these together, the number of letters in Σ is

$$5k_1 + 8k_2 + 7k_3 \leq 5k_1 + 10k_2 + 15k_3 = 5k.$$

Theorem 3 *[33] CMSR has a weak kernel of size $5k$ which implies directly an FPT algorithm running in $O(2^{3.61k}n + n^2)$ time.*

Proof. From the above discussion, we can choose to delete k letters in Σ from G'_1, G'_2. The number of choices, is hence bounded by

$$\left(\begin{array}{c} 5k \\ k \end{array} \right) \approx 2^{3.61k},$$

using Stirling's formula. For each choice, we can check whether it is valid, i.e., whether all remaining markers are in some strip in G'_1 and G'_2. This can be done in linear time if we spend $O(n^2)$ time in advance, i.e., building a correspondence between all of the identical markers in G_1, G_2. So the overall running time of the algorithm is $O(2^{3.61k}n + n^2)$ time. Note that the algorithm will report 'no solution of size k', if none of the choices leads to a valid solution. □

We comment that with the bounded search tree method, the FPT algorithm can be improved to run in $O(3^k n + n^2)$ time [36]. The details will not be reported here.

3.2 Minimum Co-Path Set

In this subsection, we study the following problem called Minimum Co-Path Set. Given a simple undirected graph G, a *co-path set* is a set S of edges in G whose removal leaves a graph in which every connected component is a path. In the Minimum Co-Path Set Problem, we need to compute a minimum co-path set in G.

The Minimum Co-Path Set Problem originates from radiation hybrid (Rh) mapping, which is a powerful technique for mapping unique DNA sequences onto chromosomes and whole genomes [9, 13, 27, 28]. In Rh mapping, chromosomes are randomly broken into small DNA fragments through gamma radiation. A (random) subset of these DNA fragments retain with healthy hamster cells and grow up to build up a hybrid cell line. This process is repeated many times and the co-retention rate of a pair of markers (labeled chromosomal loci) indicates their physical distance on the chromosome. In principle, when two markers x and y are close, the probability that x and y are broken by the gamma radiation is small, hence with a high probability they are either co-present in or co-absent from a DNA fragment.

A subset of markers that are co-present from DNA fragments is called a *cluster*. Let $V = \{1, 2, \cdots, n\}$ be a set of markers and let $\mathcal{C} = \{C_1, C_2, \cdots, C_m\}$ be a collection of clusters. The Radiation Hybrid Map Construction Problem is to compute a linear ordering of the markers in which the markers in each cluster C_i appear consecutively. In reality, a cluster might be formed with errors, so no such linear ordering might exist. In this case, one needs to remove the minimum number of clusters so that the leftover clusters admit a linear ordering. When $|C_i| = 2$ for all i, this is exactly the Minimum Co-Path Set Problem. Given a simple undirected graph $G = (V, E)$, each vertex in V corresponds to a marker, an edge $(u, v) \in E$ corresponds to a cluster $\{u, v\}$.

In [9], the Minimum Co-Path Set Problem was shown to be NP-complete [18]. The proof is by a reduction from the Hamiltonian Path problem, with each edge (u, v) being converted to a cluster $\{u, v\}$. It is easy to see that there is a Hamiltonian Path in the input graph G if and only if one has to delete exactly $|E| - n + 1$ clusters. A factor-2 approximation was also proposed in [9], which was recently improved to 10/7 [10]. (The counterpart of the Minimum Co-Path Set Problem is the well-known *Minimum Path Cover* problem [32] and will not be covered here.) Let k be the minimum number of edges deleted for the problem. We show in this subsection that the Minimum Co-Path Set Problem is in FPT; in fact, it has a linear weak kernel of size at most $5k$, hence the problem can be solved efficiently in $O(2^{3.61k}(n + k))$ time. In the following, we present the technical details.

If some connected component of G has maximum vertex degree at most 2 then the problem is trivially solvable for that component. So from now on we assume that each connected component of G has maximum vertex degree at least 3. Moreover, in the solution a single vertex could also be considered as a (degenerate) path. The following lemma is easy to prove.

Lemma 3 *There is a solution R for the minimum co-path set such that R contains only edges incident to some vertices of degree at least 3 in G.*

Proof. Assume to the contrary that a solution R contains some edge (x, y) such that both x and y have degree at most two in G. Let $G - R$ be the graph obtained from G by deleting all the edges in

R. When both x and y have degrees at most 2, if (x, y) is in R then putting it back to $G - R$ would have two possibilities: (1) make each connected component of $(G - R) \cup \{(x, y)\}$ a path, or (2) create some cycle in $(G - R) \cup \{(x, y)\}$. In case (1), it contradicts the optimality of R. In case (2), (x, y) is on some cycle in G. Hence we can find an edge (x', y') on this cycle which is incident to some vertex of degree at least 3 in G. Then we simply swap (x, y) with (x', y') in R. It is easy to see that repeating this process we can eventually have a new solution R' such that $|R'| = |R|$ and R' contains only edges incident to some vertices of degree at least 3 in G. □

Now let D be a solution for the minimum co-path set such that D contains only edges incident to some vertices of degree at least 3 in G. The above lemma implies a simple weak kernelization procedure.

1. Identify the vertices of G with degree at least 3. Let this set be $V_3(G)$.
2. Let the set of edges which are incident to some vertices in $V_3(G)$ be $E_3(G)$.
   Return $(G, E_3(G), k)$ as a weak kernel.

We have the following lemma.

Lemma 4 *The Minimum Co-Path Set Problem has a solution of size k if and only if the solution can be obtained by deleting k edges in $E_3(G)$.*

Proof. We only need to show the 'only-if' part as the other part is obvious. By Lemma 3, we do not need to include any edge in D which is incident to vertices of degree only one or two. □

It remains to show the weak kernel size (i.e., the size of $E_3(G)$). We have the following lemma.

Lemma 5 *Let $k = |D|$, then $|E_3(G)| \leq 5k$. In other words, the size of the weak kernel of the Minimum Co-Path Set Problem is $5k$.*

Proof. From Lemma 4, we know that the k edges of D can be found in $E_3(G)$. After these k edges in D are deleted from G, $G - D$ is only composed of paths, i.e., the degrees of vertices in $G - D$ are at most 2. In other words, the edges in $E_3(G) - D$ must also be incident to vertices in $G - D$ of degree at most 2 (note that these vertices originally are all in $V_3(G)$). As the k edges in D are incident to at most $2k$ vertices in $V_3(G)$, $|V_3(G)| \leq 2k$. Therefore, we have at most $4k$ edges in $E_3(G) - D$. Counting the k edges in D back, we have $|E_3(G)| = |E_3(G) - D| + |D| \leq 4k + k = 5k$.
□

With the above lemmas, it is easy to have an FPT algorithm for the Minimum Co-Path Set Problem. First, if $|V_3(G)| > 2k$ or $|E_3(G)| > 5k$ then we can simply return NO. Otherwise, among the (at most) $5k$ edges in $E_3(G)$, select all combinations of k edges to delete. For each set of k

edges selected, delete them from G and check whether the resulting graph is composed of paths only (using standard linear time graph algorithms like depth-first search). If we fail to find such a set, then return 'No solution of size k'; otherwise, just return the computed set of edges as D.

The time complexity of the algorithm is dominated by checking $\left(\begin{array}{c} 5k \\ k \end{array}\right) \approx 2^{3.61k}$ solutions. This presents an FPT algorithm for the Minimum Co-Path Set Problem which runs in $O(2^{3.61k}(n+k))$ time. We have the following theorem.

Theorem 4 *Let k be the size of the minimum co-path set. The Minimum Co-Path Set Problem has a weak kernel of size $5k$, hence can be solved in $O(2^{3.61k}(n+k))$ time.*

Similar to CMSR, we show recently, with the bounded search tree method, that the Minimum Co-Path Set Problem can be solved in $O(2.46^k(n+k))$ time. The results will be reported elsewhere. This problem is also closely related to the problem of deleting the minimum number of vertices in G so that the remaining vertices have max-degree of d (e.g., 2), which has a linear kernel for any fixed d [16].

3.3 Sorting with Minimum Unsigned Reversals

Sorting with Minimum Unsigned Reversals (SMUR) is a famous problem in computational biology, more specifically, in computational genomics. Given a genome H composed of a sequence of n distinct genes (also formulated as a permutations of n integers $\{1, 2, \cdots, n\}$), i.e., assume that $H = s_1 s_2 \cdots s_i s_{i+1} \cdots s_{j-1} s_j \cdots s_n$, a *reversal* operation on the segment $s_i s_{i+1} \cdots s_{j-1} s_j$ transforms H into $H' = s_1 s_2 \cdots s_j s_{j-1} \cdots s_{i+1} s_i \cdots s_n$. The problem Sorting with Minimum Unsigned Reversals is to use the minimum number of reversals to convert H into the identity permutation $I = 123 \cdots n$. Example: Given $H = 15342$, we can use two signed reversals to first change it to 15432 and finally to 12345.

When the genes are signed, we have a similar problem Sorting with Minimum Signed Reversals. Given a signed genome H^- composed of a sequence of n distinct (signed) genes (also formulated as a signed permutations of n integers $\{1, 2, \cdots, n\}$), i.e., $H^- = t_1 t_2 \cdots t_i t_{i+1} \cdots t_{j-1} t_j \cdots t_n$, a *signed reversal* operation on the segment $t_i t_{i+1} \cdots t_{j-1} t_j$ transforms H^- into $H'' = t_1 t_2 \cdots - t_j - t_{j-1} \cdots - t_{i+1} - t_i \cdots t_n$. The problem Sorting with Minimum Signed Reversals is to use the minimum number of signed reversals to convert H^- into the identity permutation $I = 123 \cdots n$. Example: Given $H = 1{-}534{-}2$, we can use two signed reversals to first change it to $1{-}5{-}4{-}3{-}2$ and finally to 12345. (Note that in the literature it is also acceptable to convert H^- to $-I = -n \cdots - 3 - 2 - 1$. We can enforce that H^- is converted to I by adding two auxiliary genes, i.e., $0H^-(n+1)$. This is a known trick in computational genomics.)

SMUR was shown to be NP-complete by Caprara [7] and the best approximation algorithm has a factor 1.375 [2]. However, no non-trivial FPT algorithm is known for the problem. The trivial solution is to use a bounded search tree algorithm which runs in roughly $O(k^{O(k)}n)$ time. We show below that with weak kernels, a much faster FPT algorithm can be designed.

We use Sorting with Minimum Signed Reversals as a subroutine for SMUR. Unlike SMUR, Sorting with Minimum Signed Reversals can be solved in polynomial time [21, 25, 31], with the best running time being $O(n \log n)$ [29]. Computing the minimum signed reversal distance, however, can be done in linear time [4]. Let H be the (unsigned) genome to be sorted. It is easy to see that each reversal can eliminate at most two breakpoints. (In this case a breakpoint is a 2-substring $\langle i, j\rangle$ of H such that $|j - i| \neq 1$.) Hence, if the optimal solution size is k, there would be at most $2k$ breakpoints in H. In other words, there are at most $4k$ genes which are in some breakpoints. Let H_k be the set of such (at most) $4k$ genes.

Given H, let a maximal substring B of H composed of at least two consecutive adjacencies be called a *block*, with the first and last letters called *head* and *tail* of the block respectively. (We also say that the head and the tail are *adjacent through the block* B in H.) Example: $H = \langle 0, 5, 7, 8, 10, 1, 2, 3, 4, 9, 6, 11\rangle$, $B = \langle 1, 2, 3, 4\rangle$ is a block with head 1 and tail 4. 7 and 8 are in H_7 but form an adjacency in H. 1 and 4 are adjacent through the block B in H. Following [20], there is an optimal SMUR solution for H which does not cut any block.

Let H_k^- be the set of signed genomes obtained by adding $+/-$ signs on these genes (involved in some breakpoints) in H_k. (Following [20], if two such genes in H_k are the head and tail of a block B, then all the genes in B should be given the same sign, i.e., either all positive or all negative.) It is easily seen that $|H_k^-| \leq 2^{4k}$. Moreover, we have the following lemma.

Lemma 6 *There is a solution of k unsigned reversals for sorting H if and only if the solution can be found by sorting some sequence in H_k^- with k signed reversals.*

Proof. If there is a solution of k unsigned reversals for sorting H, then we can trace these k reversals backwards and each time add signs accordingly. For example, assume that the last reversal to obtain $\langle 0, 1, 2, 3, 4, 5\rangle$ is $\underline{\langle 3, 2\rangle}$, then for sorting by signed reversals the second last signed genome is $\langle 0, 1, -3, -2, 4, 5\rangle$. It is easily seen that after repeating this process k times, we have a signed genome H'' in H_k^-. Certainly, one can sort H'' by k signed reversals.

On the other hand, if there are k signed reversals which sorts some genome in H_k^-, say H'', one can ignore the negative signs in H'' (to obtain H) and perform the same k (unsigned) reversals to sort H into I. □

Theorem 5 *Sorting with Minimum Unsigned Reversals has a weak kernel of size $4k$, hence can be solved in $O(2^{2k}n + n \log n)$ time.*

Proof. We first show a bound of $O(2^{4k}n + n \log n)$, which is straightforward from the $4k$ weak kernel. First, following Lemma 6, the weak kernelization is easy: identify all the blocks in H and return (H, H_k, k). For each possible signed genome in H_k^- (obtained from H_k by adding some negative signs), we use the $O(n)$ time algorithm in [4] to check whether it can be sorted with k signed reversals. If so, we can compute accordingly the k signed reversals using the algorithm by Swenson *et al.* [29], to obtain the k (unsigned) reversals to sort H in $O(n \log n)$ time. If no valid solution is found, we report NO. This algorithm clearly runs in $O(2^{4k}n + n \log n)$ time.

By a more detailed analysis (i.e., we do not have to try all possible ways to sign genes in H_k), the running time of the above algorithm can be improved to $O(2^{2k}n + n \log n)$ time. Now let the genes in H_k form a total of z adjacencies (possibly through some blocks). Following [20], if two such genes form an adjacency in H, obviously they have to be given the same signs, i.e., either both positive or both negative. If two such genes form an adjacency through some block B in H, all the genes in B need to have the same signs. So the total number of ways to sign genes in H_k is bounded by

$$2^z \times 2^{(4k-2z)/2-1} = 2^{2k-1}.$$

Hence we have an FPT algorithm with running time $O(2^{2k}n + n \log n)$. □

We comment that, for the related Sorting with Minimum Unsigned Translocation problem, exactly the same idea can be applied to obtain a weak kernel of size $4k$, hence an FPT algorithm with running time $O(2^{2k}n + n^2)$. The relevant details can be found in [34, 3] (or from the references therein).

4 Concluding Remarks

We formally introduce a new (somehow a previous folklore) concept called weak kernels for fixed-parameter computation and proved some interesting properties of weak kernels. We also show some interesting applications with weak kernels. We believe that for certain problems weak kernels are more flexible and possibly more powerful than the traditional kernels. This is certainly the case with our three applications, especially the famous Sorting with Minimum Unsigned Reversals (SMUR) problem. We know of no FPT algorithm which runs close to $O^*(2^{4k})$ time. It would be interesting to see more applications of weak kernels.

We feel that the running times of the best FPT algorithms for the three problems can all be further improved, maybe with some new techniques. At this point for decent k (say $k = 40$), none of them is really fast enough for practical datasets.

Finally, the Minimum Co-Path Set Problem only handles the special case of the Radiation Hybrid Map Construction Problem (i.e., when $|C_i| = 2$). It would also be interesting to tackle the general problem with exact and approximation algorithms.

Acknowledgments

This research is partially supported by NSF of China under project 60928006. We also thank Mike Fellows and Angsheng Li for several valuable comments.

References

[1] P. Bonsma, T. Brüggemann and G. Woeginger. A fast FPT algorithm for finding spanning trees with many leaves. In *Proc. 28th Intl. Symp. on Mathematical Foundations of Computer Science (MFCS'03), pages 259-268, 2003 (LNCS series, 2747).*

[2] P. Berman, S. Hannenhalli and M. Karpinski. 1.375-approximation algorithm for sorting by reversals. In *Proc. 10th European Symp. on Algorithms (ESA'02), pages 200-210, Rome, Italy, Sep, 2002 (LNCS series, 2461).*

[3] A. Bergeron, J. Mixtacki and J. Stoye. On sorting by translocation. In *Proc. 9th Intl. Conf. on Research in Comput. Molecular Biology (RECOMB'05),* pages 615-629, 2005.

[4] D. Bader, B. Moret and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *J. of Computational Biology*, 8:483–491, 2001.

[5] H. Bodlaender, R. Downey, M. Fellows and D. Hermelin. On problems without polynomial kernels. In *Proc. 35th Intl. Colloquium on Automata, Languages and Programming (ICALP'08), pages 563-574, 2008 (LNCS series, 5125).*

[6] L. Bulteau, G. Fertin and I. Rusu. Maximal strip recovery problem with gaps: hardness and approximation algorithms. *Proceedings of the 20th Annual International Symposium on Algorithms and Computation (ISAAC'09),* LNCS 5878, pages 710-719, 2009.

[7] A. Caprara. Sorting by reversals is difficult. In *Proc. 1st Intl. Conf. on Research in Comput. Molecular Biology (RECOMB'97),* pages 75-83, 1997.

[8] Z. Chen, B. Fu, M. Jiang, and B. Zhu. On recovering syntenic blocks from comparative maps. *Journal of Combinatorial Optimization*, 18:307–318, 2009.

[9] Y. Cheng, Z. Cai, R. Goebel, G. Lin and B. Zhu. The radiation hybrid map construction problem: recognition, hardness, and approximation algorithms. *Unpublished Manuscript*, 2008.

[10] Z. Chen, G. Lin and L. Wang. An approximation algorithm for the minimum co-path set problem. *Algorithmica*, to appear, 2010.

[11] V. Choi, C. Zheng, Q. Zhu, and D. Sankoff. Algorithms for the extraction of synteny blocks from comparative maps. In *Proceedings of the 7th International Workshop on Algorithms in Bioinformatics (WABI'07)*, pages 277–288, 2007.

[12] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symp. on Theory of Computing (STOC'71)*, pages 151–158, 1971.

[13] D.R. Cox, M. Burmeister, E.R. Price, S. Kim, and R.M. Myers. Radiation hybrid mapping: a somatic cell genetic method for constructing high resolution maps of mammalian chromosomes. *Science*, 250:245–250, 1990.

[14] R. Downey and M. Fellows. *Parameterized Complexity*, Springer-Verlag, 1999.

[15] M. Fellows. The lost continent of polynomial time: preprocessing and kernelization. In *Proc. 2nd Intl. Workshop on Parameterized and Exact Computation (IWPEC'06), LNCS 4169, pages 276-277, 2006.*

[16] M. Fellows, J. Guo, H. Moser and R. Niedermeier. A generalization of Nemhauser and Trotter's local optimization theorem. In *Proc. 26th Intl. Symp. on Theoretical Aspects of Computer Science (STACS'09), pages 409-420, 2009.*

[17] H. Fernau, F. Fomin, D. Lokshtanov, D. Raible, S. Saurabh and Y. Villanger. Kernel(s) for problems with no kernel: on out-trees with many leaves. In *Proc. 26th Intl. Symp. on Theoretical Aspects of Computer Science (STACS'09), pages 421-432, 2009.*

[18] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[19] J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *SIGACT News*, 38:31-45. 2007.

[20] S. Hannenhalli and P. Pevzner. To cut...or not to cut (Applications of comparative physical maps in molecular evolution). In *Proceedings of the 7th ACM-SIAM Symp. on Discrete Algorithms (SODA'96)*, pages 304–313, 1996.

[21] S. Hannenhalli and P. Pevzner. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *J. ACM*, **46**(1):1-27. 1999.

[22] M. Jiang. Inapproximability of maximal strip recovery. *Proceedings of the 20th Annual International Symposium on Algorithms and Computation (ISAAC'09)*, LNCS 5878, pages 616-625, 2009.

[23] M. Jiang. Inapproximability of maximal strip recovery, II. *Proceedings of the 4th Annual Frontiers of Algorithmics Workshop (FAW'10)*, to appear, 2010.

[24] D. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Sys. Sciences*, 9:256-278, 1974.

[25] H. Kaplan, R. Shamir and R. Tarjan. A faster and simpler algorithm for sorting signed permutations by reversals. *SIAM J. Comput.*, 29:880-892, 1999.

[26] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher (eds.), *Complexity of Computer Computations*, Plenum Press, NY, pages 85–103, 1972.

[27] C.W. Richard, D.A. Withers, T.C. Meeker, S. Maurer, G.A., Evans, R.M. Myers, and D.R. Cox. A radiation hybrid map of the proximal long arm of human chromosome 11, containing the multiple endocrine neoplasia type 1 (MEN-1) and bcl-1 disease loci. *American J. of Human Genetics*, 49:1189-1196, 1991.

[28] D. Slonim, L. Kruglyak, L. Stein, and E. Lander. Building human genome maps with radiation hybrids. *J. of Computational Biology*, 4:487–504, 1997.

[29] K. Swenson, V. Rajan, Y. Lin, and B. Moret. Sorting signed permutations by inversions in $O(n \log n)$ time. In *Proc. RECOMB'09*, LNCS 5541, pages 386–399, 2009.

[30] R. Tarjan and A. Trojanowski. Finding a maximum independent set. *SIAM J. Comput.*, 6:537-546, 1977.

[31] E. Tannier and M-F. Sagot. Sorting by reversals in subquadratic time. In *Proc. 15th Symp. Combinatorial Pattern Matching (CPM'04), Istanbul, Turkey, pages 1-13, July, 2004 (LNCS series, 3109).*

[32] S. Vishwanathan. An approximation algorithm for the asymmetric travelling salesman problem with distance one and two. *Information Processing Letters*, 44:297–302, 1992.

[33] L. Wang and B. Zhu. On the tractability of maximal strip recovery. *J. of Computational Biology*, to appear, 2010. (An earlier version appeared in TAMC'09.)

[34] L. Wang, D. Zhu, X. Liu and S. Ma. An $O(n^2)$ algorithm for signed translocation. *J. Comput. Sys. Sciences*, 70:284-299, 2005.

[35] C. Zheng, Q. Zhu, and D. Sankoff. Removing noise and ambiguities from comparative maps in rearrangement analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4:515–522, 2007.

[36] B. Zhu. Efficient exact and approximate algorithms for the complementary of maximal strip recovery. *Proc. of AAIM'10*, to appear, LNCS series 6124, 2010.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de