Lower Bounds and Parameterized Approach for Longest Common Subsequence

Xiuzhen Huang

Department of Computer Science,
Arkansas State University,
P.O. Box 9, State University,
Arkansas 72467, USA
xzhuang@csm.astate.edu

Abstract. In this paper, different parameterized versions of the LONGEST COMMON SUBSEQUENCE (LCS) problem are extensively investigated and computational lower bound results are derived based on current research progress in parameterized computation. For example, with the number of sequences as the parameter k, the problem is unlikely to be solvable in time $f(k)n^{o(k)}$, where n is the length of each sequence and f is any recursive function. The lower bound result is asymptotically tight in consideration of the dynamic programming approach of time $O(n^k)$. Computational lower bounds for polynomial-time approximation schemes (PTAS) for the LCS problem are also derived. It is shown that the LCS problem has no PTAS of time $f(1/\epsilon)n^{o(1/\epsilon)}$ for any recursive function f, unless all SNP problems are solvable in subexponential time. Compared with former results on this problem, this result has its significance. Finally a parameterized approach for the LCS problem is discussed, which is more efficient than the dynamic programming approach, especially when applied to large scale sequences.

1 Introduction

A string s is a subsequence of a string s' if s can be obtained from s' by deleting some characters in s'. For example, "ac" is a subsequence of "atcgt". Given a set of strings over an alphabet Σ, the LONGEST COMMON SUBSEQUENCE problem is to find a common subsequence that has the maximum length. The alphabet Σ may be of fixed size or of unbounded size. The LONGEST COMMON SUBSEQUENCE (LCS) problem is a well-known optimization problem because of its applications, especially in bioinformatics. The fixed alphabet version of the problem is of particular interest considering the importance of sequence comparison (e.g. multiple sequence alignment) in the fixed size alphabet world of DNA and protein sequences. (Note that in computational biology, DNA sequences are in a four-letter alphabet, and protein sequences are in a twenty-letter alphabet).

D.Z. Chen and D.T. Lee (Eds.): COCOON 2006, LNCS 4112, pp. 136–145, 2006.

We study the LONGEST COMMON SUBSEQUENCE problem in parameterized computation in this paper. We first give a brief review on parameterized complexity theory and some recent progress on parameterized intractability. A *parameterized problem* Q is a decision problem consisting of instances of the form (x, k), where the integer $k \geq 0$ is called the *parameter*. The parameterized problem Q is *fixed-parameter tractable* [15] if it can be solved in time $f(k)|x|^{O(1)}$, where f is a recursive function[1]. Certain NP-hard parameterized problems, such as VERTEX COVER, are fixed-parameter tractable, and hence can be solved practically for small parameter values [10]. On the other hand, the inherent computational difficulty for solving many other NP-hard parameterized problems with even small parameter values has motivated the theory of *fixed-parameter intractability* [15]. The W-hierarchy $\bigcup_{t\geq 1} W[t]$ has been introduced to characterize the inherent level of intractability for parameterized problems. Examples of $W[1]$-hard problems include problems such as CLIQUE and DOMINATING SET. It has become commonly accepted that no $W[1]$-hard problem can be solved in time $f(k)n^{O(1)}$ for any function f, i.e., $W[1] \neq FPT$. $W[1]$-hardness has served as the hypothesis for fixed-parameter intractability.

Based on the W[1]-hardness of the CLIQUE algorithm, computational intractability of problems in computational biology has been derived [2,3,16,17,23,25]. For example, in [25], the author point out that "unless an unlikely collapse in the parameterized hierarchy occurs, this (This refers to the results proved in [25] that the problems LONGEST COMMON SUBSEQUENCE and SHORTEST COMMON SUPERSEQUENCE are $W[1]$-hard) rules out the existence of exact algorithms with running time $f(k)n^{O(1)}$ (i.e., exponential only in k) for those problems. This does not mean that there are no algorithms with much better asymptotic time-complexity than the known $O(n^k)$ algorithms based on dynamic programming, e.g., algorithms with running time $n^{\sqrt{k}}$ are not deemed impossible by our results."

Recent investigation in [7,8] has derived stronger computational lower bounds for well-known NP-hard parameterized problems. For example, for the CLIQUE problem, which asks if a given graph of n vertices has a clique of size k, it is proved that unless an unlikely collapse occurs in parameterized complexity theory, the problem is not solvable in time $f(k)n^{o(k)}$ for *any* function f. Note that this lower bound is asymptotically tight in the sense that the trivial algorithm that enumerates all subsets of k vertices in a given graph to test the existence of a clique of size k runs in time $O(n^k)$. Based on the hardness of the CLIQUE problem, lower bound results for a number of computational biology problems have been derived [18,9].

In this paper, we extensively investigate different parameterized versions of the LONGEST COMMON SUBSEQUENCE problem. Our results for the problem strengthen the results in the literature (such as [25]) significantly and advance our understanding on the complexity of the problems.

[1] In this paper, we always assume that complexity functions are "nice" with both domain and range being non-negative integers and the values of the functions and their inverses can be easily computed.

2 Terminologies in Approximation

We provide some basic terminologies for studying approximation algorithms and its relationship with parameterized complexity. For a reference of the theory of approximation, the readers are referred to [1].

An *NP optimization problem* Q is a 4-tuple (I_Q, S_Q, f_Q, opt_Q), where

1. I_Q is the set of input instances. It is recognizable in polynomial time;
2. For each instance $x \in I_Q$, $S_Q(x)$ is the set of feasible solutions for x, which is defined by a polynomial p and a polynomial time computable predicate π (p and π only depend on Q) as $S_Q(x) = \{y : |y| \leq p(|x|) \text{ and } \pi(x, y)\}$;
3. $f_Q(x, y)$ is the objective function mapping a pair $x \in I_Q$ and $y \in S_Q(x)$ to a non-negative integer. The function f_Q is computable in polynomial time;
4. $opt_Q \in \{\max, \min\}$. Q is called a *maximization problem* if $opt_Q = \max$, and a *minimization problem* if $opt_Q = \min$.

An *optimal solution* y_0 for an instance $x \in I_Q$ is a feasible solution in $S_Q(x)$ such that $f_Q(x, y_0) = opt_Q\{f_Q(x, z) \mid z \in S_Q(x)\}$. We will denote by $opt_Q(x)$ the value $opt_Q\{f_Q(x, z) \mid z \in S_Q(x)\}$.

An algorithm A is an *approximation algorithm* for an NP optimization problem $Q = (I_Q, S_Q, f_Q, opt_Q)$ if, for each input instance x in I_Q, A returns a feasible solution $y_A(x)$ in $S_Q(x)$. The solution $y_A(x)$ has an *approximation ratio* $r(n)$ if it satisfies the following condition:

$$opt_Q(x)/f_Q(x, y_A(x)) \leq r(|x|) \text{ if } Q \text{ is a maximization problem}$$
$$f_Q(x, y_A(x))/opt_Q(x) \leq r(|x|) \text{ if } Q \text{ is a minimization problem}$$

The approximation algorithm A has an *approximation ratio* $r(n)$ if for any instance x in I_Q, the solution $y_A(x)$ constructed by the algorithm A has an approximation ratio bounded by $r(|x|)$.

Definition 1. *An NP optimization problem Q has a polynomial-time approximation scheme (PTAS) if there is an algorithm A_Q that takes a pair (x, ϵ) as input, where x is an instance of Q and $\epsilon > 0$ is a real number, and returns a feasible solution y for x such that the approximation ratio of the solution y is bounded by $1 + \epsilon$, and for each fixed $\epsilon > 0$, the running time of the algorithm A_Q is bounded by a polynomial of $|x|$.*

An NP optimization problem Q can be parameterized in a natural way as follows. The following definition offers the possibility to study the relationship between the approximability and the parameterized complexity of NP optimization problems.

Definition 2. *Let $Q = (I_Q, S_Q, f_Q, opt_Q)$ be an NP optimization problem. The parameterized version of Q is defined as follows:*

(1) If Q is a maximization problem, then the parameterized version of Q is defined as $Q_{\geq} = \{(x, k) \mid x \in I_Q \wedge opt_Q(x) \geq k\}$;

(2) If Q is a minimization problem, then the parameterized version of Q is defined as $Q_{\leq} = \{(x, k) \mid x \in I_Q \wedge opt_Q(x) \leq k\}$.

3 Lower Bound Results for LCS

In the following we derive the lower bounds for the exact algorithms for the parameterized versions of the LONGEST COMMON SUBSEQUENCE (LCS) problem. We also extend the techniques and derive the lower bounds for the approximation algorithms for the optimization versions of the problem.

3.1 Formal Problem Definitions

Several parameterized versions of the LCS problem are discussed in [2,3,17,25]. We present the four parameterized versions of the problem.

The **LCS-k** problem:
Instance: a set $S = \{s_1, s_2, ..., s_k\}$ of strings over an alphabet Σ, and an integer $\lambda > 0$, where the alphabet Σ is of unbounded size.
Parameter: k.
Question: is there a string $s \in \Sigma^*$ of length λ, which is a subsequence of each string in S?
The **FLCS-k** problem:
Instance: a set $S = \{s_1, s_2, ..., s_k\}$ of strings over an alphabet Σ, and an integer $\lambda > 0$, where the alphabet Σ is of fixed size.
Parameter: k.
Question: is there a string $s \in \Sigma^*$ of length λ, which is a subsequence of each string in S?
The **LCS-λ** problem:
Instance: a set $S = \{s_1, s_2, ..., s_k\}$ of strings over an alphabet Σ, and an integer $\lambda > 0$, where the alphabet Σ is of unbounded size.
Parameter: λ.
Question: is there a string $s \in \Sigma^*$ of length λ, which is a subsequence of each string in S?
The **FLCS-λ** problem:
Instance: a set $S = \{s_1, s_2, ..., s_k\}$ of strings over an alphabet Σ, and an integer $\lambda > 0$, where the alphabet Σ is of fixed size.
Parameter: λ.
Question: is there a string $s \in \Sigma^*$ of length λ, which is a subsequence of each string in S?

The following results on the parameterized complexity of these parameterized problems are known:

- The LCS-k problem is W[t]-hard for $t \geq 1$ [3].
- The FLCS-k problem is W[1]-hard [25].
- The LCS-λ problem is W[2]-hard [3].
- The FLCS-λ problem is in FPT [25].

In particular, we are interested in the FLCS-k problem and the LCS-λ problem, which we discuss in the following sections.

3.2 FLCS-k

In [25], the FLCS-k problem is proved to be $W[1]$-hard. Unless $W[1] = \text{FPT}$, for the FLCS-k problem, the $W[1]$-hardness result rules out the existence of algorithms of time $f(k)n^{O(1)}$ for any function f, where k is the number of strings. In the conclusion of [25], the author pointed out that the $W[1]$-hardness of FLCS-k does not exclude the possibility of having an algorithm of time, say $O(n^{\sqrt{k}})$, which is much more efficient than the $O(n^k)$ time dynamic programming algorithm for the FLCS-k problem.

However, it is proved that

Theorem 1 ([9]). *The FLCS-k problem has no algorithm of time $f(k)n^{o(k)}$ for any function f, unless all SNP problems are solvable in subexponential time.*

Interested readers are referred to [9,18] for a detailed proof of this result.

The class SNP introduced by Papadimitriou and Yannakakis [22] contains many well-known NP-hard problems including, for any fixed integer $q \geq 3$, CNF q-SAT, q-COLORABILITY, q-SET COVER, and VERTEX COVER, CLIQUE, and INDEPENDENT SET [19]. It is commonly believed that it is unlikely that all problems in SNP are solvable in subexponential time[2].

We define an optimization problem FLCS-k_{opt} and its corresponding parameterized problem FLCS'-k.

> The **FLCS-k_{opt}** problem:
> given a set $S = \{s_1, s_2, ..., s_l\}$ of strings over a fixed alphabet Σ, and an integer $\lambda > 0$, try to find a string $s \in \Sigma^*$ of length λ maximizing the size of a subset S' of S, such that s is a common subsequence of all the strings in S'.

By our definition, the parameterized version of the optimization problem FLCS-k_{opt} is

> The **FLCS'-k** problem:
> Instance: given a set $S = \{s_1, s_2, ..., s_l\}$ of strings over a fixed alphabet Σ, and an integer $\lambda > 0$.
> Parameter: an integer k, $0 < k \leq l$.
> Question: is there a string $s \in \Sigma^*$ of length λ such that s is a common subsequence of at least k strings in the set S?

From the definitions of the two parameterized problems FLCS-k and FLCS'-k, we can see that FLCS-k is a special case of FLCS'-k. There is a trivial linear fpt-reduction from FLCS-k to FLCS'-k: given an instance I_1 of FLCS-k, $I_1 = (S_1 = \{s_1, s_2, ..., s_k\}, \lambda$ and the parameter $k)$, we build an instance I_2 of FLCS'-k, $I_2 = (S_2 = \{s_1, s_2, ..., s_k\}, \lambda$ and the parameter $k)$, which asks if there is a string $s \in \Sigma^*$ of length λ that is a common subsequence of at least k strings

[2] A recent result showed the equivalence between the statement that all SNP problems are solvable in subexponential time, and the collapse of a parameterized class called *Mini*[1] to *FPT* [14].

(i.e., all strings) in the set S_2. Obviously, the instance I_2 is a yes-instance for the problem FLCS'-k if and only if the instance I_1 is a yes-instance for the problem FLCS-k, .

Theorem 2 ([8,9]). *Suppose that a problem Q_1 has no algorithm of time $f(k)n^{o(k)}$ for any function f, and that Q_1 is linear fpt-reducible to Q_2. Then the problem Q_2 has no algorithm of time $f'(k)n^{o(k)}$ for any function f'.*

By the above linear fpt-reduction, Theorem 1 and Theorem 2, we have

Lemma 1. *The FLCS'-k problem has no algorithm of time $f(k)n^{o(k)}$ for any function f, unless all SNP problems are solvable in subexponential time.*

Theorem 3 ([8,9]). *Let Q be an NP optimization problem. If the parameterized version of Q has no algorithm of time $f(k)n^{o(k)}$, then Q has no PTAS of running time $f(1/\epsilon)n^{o(1/\epsilon)}$ for any function f, unless all problems in SNP are solvable in subexponential time.*

Therefore, by Lemma 1 and Theorem 3, we have

Theorem 4. *The FLCS-k_{opt} problem has no PTAS of time $f(1/\epsilon)n^{o(1/\epsilon)}$ for any function f, unless all SNP problems are solvable in subexponential time.*

3.3 LCS-λ

The LCS-λ problem is proved to be $W[2]$-hard in [2,3]. Therefore, unless $W[2]$ = FPT, for the LCS-λ problem, there is no algorithm of time $f(\lambda)n^{O(1)}$ for any function f. We prove

Theorem 5. *The LCS-λ problem has no algorithm of time $f(\lambda)n^{o(\lambda)}$ for any function f, unless all SNP problems are solvable in subexponential time.*

Proof. We first give an linear fpt-reduction from DOMINATING SET to the LCS-λ problem. Based on the linear fpt-reduction, the lower bound result for DOMINATING SET [8] and Theorem 2, the theorem is proved.

The fpt-reduction from DOMINATING SET to the LCS-λ problem in [3] for proving the LCS-λ problem is $W[2]$-hard is essentially an linear fpt-reduction.

Given a graph $G = (V, E)$, $|V| = n$, and a parameter λ, and suppose an ascending order of the vertices $\{u_1, u_2, ..., u_n\}$ of G, we will construct a set S of strings such that they have a common subsequence of length λ if and only if G has a dominating set of size λ. The alphabet is $\Sigma = \{a[i, j] : 1 \leq i \leq \lambda, 1 \leq j \leq n\}$. We use the notations: $\Sigma_i = \{a[i, j] : 1 \leq j \leq n\}$, $\Sigma[t, u] = \{a[i, j] : (i \neq t)$ or $(i = t$ and $j \in N[u])\}$.

If $\Gamma \subseteq \Sigma$, let $(\uparrow \Gamma)$ be the string of length $|\Gamma|$ which consists of one occurrence of each symbol in Γ in ascending order, and let $(\downarrow \Gamma)$ be the string of length $|\Gamma|$ which consists of one occurrence of each symbol in Γ in descending order.

The set S consists of the following strings.

Control strings:

$X_1 = \Pi_{i=1}^{\lambda}(\uparrow \Sigma_i)$,
$X_2 = \Pi_{i=1}^{\lambda}(\downarrow \Sigma_i)$.
Check strings: For $u = 1, ..., n$:
$X_u = \Pi_{i=1}^{\lambda}(\uparrow \Sigma[i, u])$,

We observe that any sequence C of length λ that is a common subsequence of both control strings must consist of exactly one symbol from each Σ_i in ascending order. For such a sequence C we may associate the set V_c of vertices represented by C: if $C = a[1, u_1]...a[\lambda, u_\lambda]$, then $V_c = \{u_i : 1 \leq i \leq \lambda\} = \{x : \exists i \ a[i, x] \in C\}$.

We will prove that if C is also a subsequence of the check strings $\{X_u\}$, then V_c is a dominating set in G. Let $u \in V(G)$ and fix a substring C_u of X_u, with $C_u = C$. We have the fact [3]:

> Fact. For some index j, $1 \leq j \leq \lambda$, the symbol $a[j, u_j]$ occurs in the $(\uparrow \Sigma[j, u])$ portion of X_u, thus $u_j \in N[u]$ by the definition of $\Sigma[j, u]$.

By the above fact, if C is a subsequence of the control and check strings, then every vertex of G has a neighbor in V_c, that is, V_c is a dominating set in G.

On the other hand, if $D = \{u_1, .., u_\lambda\}$ is a dominating set in G with $u_1 < ... < u_\lambda$, then the sequence $C = a[1, u_1]...a[\lambda, u_\lambda]$ is easily seen to be a common subsequence of the strings in S.

The reduction from DOMINATING SET to LCS-λ is an linear fpt-reduction. □

Formally, we give the definition of the optimization problem LCS-λ_{opt}.

> The **LCS-λ_{opt}** problem:
> given a set $S = \{s_1, s_2, ..., s_k\}$ of strings over an alphabet Σ of unbounded size, try to find a string $s \in \Sigma^*$ of maximum length such that s is a common subsequence of all the strings in S.

By our definition, the parameterized version of the optimization problem LCS-λ_{opt} is

> The **LCS'-λ** problem:
> Instance: given a set $S = \{s_1, s_2, ..., s_k\}$ of strings over an alphabet Σ of unbounded size.
> Parameter: an integer $\lambda > 0$.
> Question: is there a string $s \in \Sigma^*$ of length at least λ such that s is a common subsequence of all strings in the set S?

Since that there is a string s of length at least λ such that s is a common subsequence of all strings in S is equivalent to that there is a string s of length exactly λ such that s is a common subsequence of all strings in S, the two problems LCS-λ and LCS'-λ are equivalent. By Theorem 5, the problem LCS'-λ has no algorithm of time $f(\lambda)n^{o(\lambda)}$ for any function f, unless all SNP problems are solvable in subexponential time. This result plus Theorem 3 gives us the following theorem:

Theorem 6. *The LCS-λ_{opt} problem has no PTAS of time $f(1/\epsilon)n^{o(1/\epsilon)}$ for any function f, unless all SNP problems are solvable in subexponential time.*

In [20], the authors showed that the LCS-λ_{opt} problem is inherently hard to approximate in the worst case. In particular, they proved that there exists a constant $\delta > 0$ such that, the LCS-λ_{opt} has no polynomial time approximation algorithm with performance ratio n^{δ}, unless P = NP. It is obvious to see that this lower bound holds only when the objective function value λ is larger than n^d for a constant $d > 0$. In particular, the lower bound result in [20] does not apply to the case when the value of λ is small. For example, in case $\lambda = n^{\delta}$, a trivial common subsequence of length one is a ratio-n^{δ} approximation solution. This implies that for the LCS problem, when the length λ of the common subsequence is a small function of n, no strong lower bound result as that of [20] has been derived.

On the other hand, our lower bound result in Theorem 6 for the LCS problem can be applied when the length of the common subsequence λ is any small function of the length n of each string.

4 Parameterized Approach for LCS

Given k sequences with each sequence of length n, we discuss in this section a parameterized approach, which choose a proper parameter, the diagonal band width b. The time complexity of the approach is $O(b * n^{(k-1)})$.

The parameterized approach for finding the longest common subsequence of two given sequences is of time $O(bn)$, where n is the length of the given sequence, b is the parameter, the value of the diagonal band width. This is a great improvement over the well known dynamic programming approach of time $O(n^2)$. Especially when the length of the given sequence n is very large and the two given sequences are very similar, the parameterized approach with a small value of the diagonal band width b can find the optimal solution more efficiently.

The banded alignment idea has been investigated in [6], but the parameterized approach here incorporates the idea of how to guarantee to find the optimal solution, which is discussed in [26]. To illustrate the basic idea of the parameterized approach, consider the case of two given sequences s_1 and s_2 with the same length n. The well known dynamic programming approach for solving the LCS problem is to build a two dimensional table where each entry represents the length of the longest common subsequence between the corresponding prefix of s_1 and the corresponding prefix of s_2 [11]. There are n^2 entries of the two dimensional table. Consider a diagonal band with width b of entries starting from the middle diagonal. The basic idea of the parameterized approach is to ignore entries outside the diagonal band. If an alignment goes outside of the diagonal band with width b, it is easy to see that the corresponding longest common subsequence cannot have a length of more than $n - b$. This is because the search loses one pair of match each time it moves one entry away from the diagonal. Therefore, if the search stays within the diagonal band with width b and finally gets a common subsequence of length at least $n - b$, it is guaranteed that this solution is optimal. That is, it finds the longest common subsequence of the two given sequences s_1 and s_2. Since this parameterized approach needs to fill up a

band with width b of the two dimensional table, it takes linear time $O(bn)$, with b as the parameter.

Our experiment results show the efficiency of the parameterized approach. Especially when the two given sequences are very similar, one could pick a relatively small value for the band b in order to achieve the optimal solution, i.e., the longest common subsequences of the given two sequences.

5 Summary

In this paper computational lower bounds on the running time of the algorithms for different parameterized versions of the LONGEST COMMON SUBSEQUENCE (LCS) problem are extensively investigated. It is proved that the problem FLCS-k is unlikely to have an algorithm of time $f(k)n^{o(k)}$, where n is the length of the sequence, k is the total number of sequences and f is any recursive function. In consideration of the known upper bound of $O(n^k)$, we point out that the lower bound result is asymptotically tight. Computational lower bounds for polynomial-time approximation schemes (PTAS) for the optimization versions of the LCS problem are also derived. We then discuss a parameterized approach for the problem. Compared with the well known dynamic programming approach, the parameterized approach is much more efficient, especially when it is applied to find the longest common subsequence of very large scale sequences, which is common in sequence comparisons in bioinformatics.

References

1. G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, *Complexity and Approximation, Combinatorial Optimization Problems and Their Approximability Properties*, New York: Springer-Verlag, (1999).
2. H. L. Bodlaender, R. G. Downey, M. R. Fellows, M. T. Hallett, and H. T. Wareham, Parameterized complexity analysis in computational biology, *Computer Applications in the Biosciences*, vol. 11, pp. 49-57, (1995).
3. H. L. Bodlaender, R. G. Downey, M. R. Fellows, and H. T. Wareham, The parameterized complexity of sequence alignment and consensus, *Theoretical Computer Science*, vol. 147, pp. 31-54, (1995).
4. L. Cai and J. Chen, On fixed-parameter tractability and approximability of NP optimization problems, *Journal Of Computer and System Sciences*, vol. 54, pp. 465-474, (1997).
5. M. Cesati and L. Trevisan, On the efficiency of polynomial time approximation schemes, *Information Processing Letters*, vol. 64, pp. 165-171, (1997).
6. K. M. Chao, W. R. Pearson, and W. Miller, Aligning two sequences within a specific diagonal band, *Computer Applications in the Biosciences*, vol. 8, pp. 481-487, (1992).
7. J. Chen, B. Chor, M. Fellows, X. Huang, D. Juedes, I. Kanj, and G. Xia, Tight lower bounds for parameterized NP-hard problems, *Information and Computation*, vol. 201, pp. 216-231, (2005).

8. J. Chen, X. Huang, I. Kanj, and G. Xia, Linear FPT reductions and computational lower bounds, in *Proc. of the 36th ACM Symposium on Theory of Computing*, pp. 212-221, (2004).
9. J. Chen, X. Huang, I Kanj and G. Xia, W-hardness linear FPT-reductions: structural properties and further applications, in *proceedings of the Eleventh International Computing and Combinatorics Conference (COCOON 2005), Lecture Notes in Computer Science*, vol. 3595, pp. 975-984, (2005).
10. J. Chen, I. Kanj, and W. Jia, Vertex Cover: Further observations and further improvements, *Journal of Algorithms*, vol. 41, pp. 280-301, (2001).
11. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, Second Edition, MIT Press, (2001).
12. X. Deng, G. Li, Z. Li, B. Ma, and L. Wang, A PTAS for distinguishing (sub)string selection, *Lecture Notes in Computer Science*, vol. 2380, pp. 740-751, (2002).
13. X. Deng, G. Li, Z. Li, B. Ma, and L. Wang, Genetic design of drugs without side-effects, *SIAM Journal on Computing*, vol. 32, pp. 1073-1090, (2003).
14. R. G. Downey, V. Estivill-Castro, M. R. Fellows, E. Prieto, and F. A. Rosamond, Cutting Up is Hard to Do: the Parameterized Complexity of k-Cut and Related Problems, Electr. Notes Theor. Comput. Sci. 78: (2003).
15. R. Downey and M. Fellows, *Parameterized Complexity*, Springer, New York, (1999).
16. M. Fellows, J. Gramm, and R. Niedermeier, Parameterized intractability of motif search problems, *Lecture Notes in Computer Science*, vol. 2285, pp. 262-273, (2002).
17. M. Hallett, *An Integrated Complexity Analysi of Problems for Computational Biology*, Ph.D. Thesis, University of Victoria, (1996).
18. X. Huang, *Parameterized Complexity and Polynomial-time Approximation Schemes*, Ph.D. Dissertation, Texas A&M University, (2004).
19. R. Impagliazzo, R. Paturi, and F. Zane, Which problems have strongly exponential complexity? *Journal Of Computer and System Sciences*, vol. 63, pp. 512-530, (2001).
20. T. Jiang and M. Li, On the approximation of shortest common supersequence and longest common subsequences, *SIAM Journal on Computing*, vol. 24, pp. 1112-1139, (1995).
21. M. Li, B. Ma, and L. Wang, On the closest string and substring problems, *Jounal of the ACM*, vol. 49, pp. 157-171, (2002).
22. C. Papadimitriou and M. Yannakakis, Optimization, approximation, and complexity classes, *Journal Of Computer and System Sciences*, vol. 43, pp. 425-440, (1991).
23. C. Papadimitriou and M. Yannakakis, On limited nondeterminism and the complexity of VC dimension, *Journal Of Computer and System Sciences*, vol. 53, 161-170, (1996).
24. C. Papadimitriou and M. Yannakakis, On the complexity of database queries, *Journal Of Computer and System Sciences*, vol. 58, pp. 407-427, (1999).
25. K. Pietrzak, On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems, *Journal Of Computer and System Sciences*, vol. 67, pp. 757-771, (2003).
26. S.-H. Sze, *Lectures notes of Special Topics in Computational Biology*, Texas A&M University, (2002).