BMC Bioinformatics

BioMed Central

Research

Maximum common subgraph: some upper bound and lower bound results

Xiuzhen Huang*[1], Jing Lai[2] and Steven F Jennings[3]

Address: [1]Department of Computer Science, Arkansas State University, State University, Arkansas 72467, USA, [2]Department of Applied Science, University of Arkansas at Little Rock, Little Rock, Arkansas 72204, USA and [3]Department of Information Science, University of Arkansas at Little Rock, Little Rock, Arkansas 72204, USA

Email: Xiuzhen Huang* - xzhuang@csm.astate.edu; Jing Lai - jxlai@ualr.edu; Steven F Jennings - sfjennings@ualr.edu

* Corresponding author

from Symposium of Computations in Bioinformatics and Bioscience (SCBB06) in conjunction with the International Multi-Symposiums on Computer and Computational Sciences 2006 (IMSCCS|06)
Hangzhou, China. June 20–24, 2006

Published: 12 December 2006

BMC Bioinformatics 2006, **7**(Suppl 4):S6 doi:10.1186/1471-2105-7-S4-S6

Abstract

Background: Structure matching plays an important part in understanding the functional role of biological structures. Bioinformatics assists in this effort by reformulating this process into a problem of finding a maximum common subgraph between graphical representations of these structures. Among the many different variants of the maximum common subgraph problem, the *maximum common induced subgraph* of two graphs is of special interest.

Results: Based on current research in the area of parameterized computation, we derive a new lower bound for the exact algorithms of the maximum common induced subgraph of two graphs which is the best currently known. Then we investigate the upper bound and design techniques for approaching this problem, specifically, reducing it to one of finding a maximum clique in the product graph of the two given graphs. Considering the upper bound result, the derived lower bound result is asymptotically tight.

Conclusion: Parameterized computation is a viable approach with great potential for investigating many applications within bioinformatics, such as the maximum common subgraph problem studied in this paper. With an improved hardness result and the proposed approaches in this paper, future research can be focused on further exploration of efficient approaches for different variants of this problem within the constraints imposed by real applications.

Background

Introduction

Of the many challenging problems related to understanding the biological function of DNA, RNA, proteins, and metabolic and signalling pathways, one of the most important is comparing the structure of different molecules. The hypothesis is that structure determines function and therefore it should follow that molecules with similar structure should have similar function. Evaluating the similarity of structures can be reduced to a comparison of a set of abstracted graphs if the biological structures can be abstracted as graphs.

Using bioinformatic techniques, biological structure matching can be formulated as a problem of finding the *maximum common subgraph*. The solution to this problem has important practical applications in many areas of bioinformatics as well as in other areas, such as pattern recognition and image processing [1-3]. For example, protein threading, an effective method to predict protein tertiary structure [4-8], and RNA structural homology searching, a method for annotating and identifying new non-coding RNAs [9-12], both align a target structure against structure templates in a template database.

Song *et al* [13] makes the following definitions and proposes the following graphical models for RNA structural homology searching: A *structural unit* in a biopolymer sequence is a stretch of contiguous residues (nucleotides or amino acids). A *non-structural stretch* between two consecutive structural units is called a *loop*. A structure of the sequence is characterized by *interactions* among structural units. For example, structural units in a tertiary protein are α helices and β strands, called *cores*. Given a biopolymer sequence, a *structure graph* H = (V, E, A) can be defined such that each *vertex* in V(H) represents a structural unit, each *edge* in E(H) represents the interaction between two structural units, and each *arc* in A(H) represents the loop ended by two structural units. Similarly, the target sequence can also be represented as a *mixed graph* G, called a *sequence graph*. Based on the graphical representations, the structure-sequence alignment problem can be formulated as the problem of finding in the sequence graph G a subgraph *isomorphic* to the structure graph H such that the *objective function* optimizes the alignment score.

Problem Definition

Throughout this paper, we will use the basic definitions and terminology from [1]: All graphs are simple, undirected graphs. Two graphs are *isomorphic* if there is a one-to-one correspondence between their vertices and there is an edge between two vertices in one graph if and only if there is an edge between the two corresponding vertices in the other graph. If edge (u, v) is an edge connecting u and v, then an *induced subgraph* G' of a graph G = (V, E) consists of a vertex subset $V' \subseteq V$ and for all edges $(u, v) \in E$ where $u, v \in V'$. A graph G_{12} is a *common induced subgraph* of two given graphs G_1 and G_2 if G_{12} is isomorphic to one induced subgraph G'_1 of G_1 as well as one induced subgraph G'_2 of G_2. A *maximum common induced subgraph* (MCIS) of two given graphs G_1 and G_2 is the common induced subgraph G_{12} with the maximum number of vertices. Similarly, the *maximum common edge subgraph* (MCES) is a subgraph with the maximum number of edges common to the two given graphs. The MCIS (or MCES) between two graphs can be further divided into a connected case and a disconnected case. All the different cases of the problem are useful within different biological contexts.

Figure 1 gives an illustration of MCIS of two graphs. In this figure, the maximum common induced subgraph of G_1 and G_2 contains four vertices (2, 3, 4 and 5) and the maximum common edge subgraph of them involves five vertices (1 through 5).

MCES can be transformed into a formulation of MCIS. Interested readers are referred to [1] for details of the transformation. Here we focus on the maximum common induced subgraph (MCIS) problem. For convenience, we call it the maximum common subgraph problem.

The maximum common subgraph problem is NP-complete [14] and therefore polynomial-time algorithms for it do not exist unless P = NP. In fact, the maximum common subgraph problem is APX-hard [15] which means that it has no constant ratio approximation algorithms. This problem is a famous combinatorial intractable problem. Approaches for the maximum common subgraph problem and different variants of this problem are intensively studied in the literature [1].

In this paper, we derive a strong lower bound result for the maximum common subgraph problem in the light of the current research progress in the research area of parameterized computation. We then design the approaches for addressing this problem.

Methods

Parameterized Computation and Recent Progress on Parameterized Intractability

Many problems with important real-world applications in life science are NP-hard within the context of the theory of NP-completeness. This excludes the possibility of solving them in polynomial time unless P = NP. For example, the problems of cleaning up data, aligning multiple sequences, finding the closest string, and identifying the maximum common substructure are all famous NP-hard problems in bioinformatics [16-18,1]. A number of approaches have been proposed in dealing with these NP-hard problems. For example, the highly-acclaimed approximation approach [19] tries to come up with a "good enough" solution in polynomial time instead of an optimal solution for an NP-hard optimization problem [20-23].

The theory of parameterized computation [17] is a newly developed approach introduced to address NP-hard problems with *small* parameters. It tries to give exact algorithms for an NP-hard problem when its natural parameter is small (even if the problem size is big). A *parameterized problem* Q is a decision problem consisting

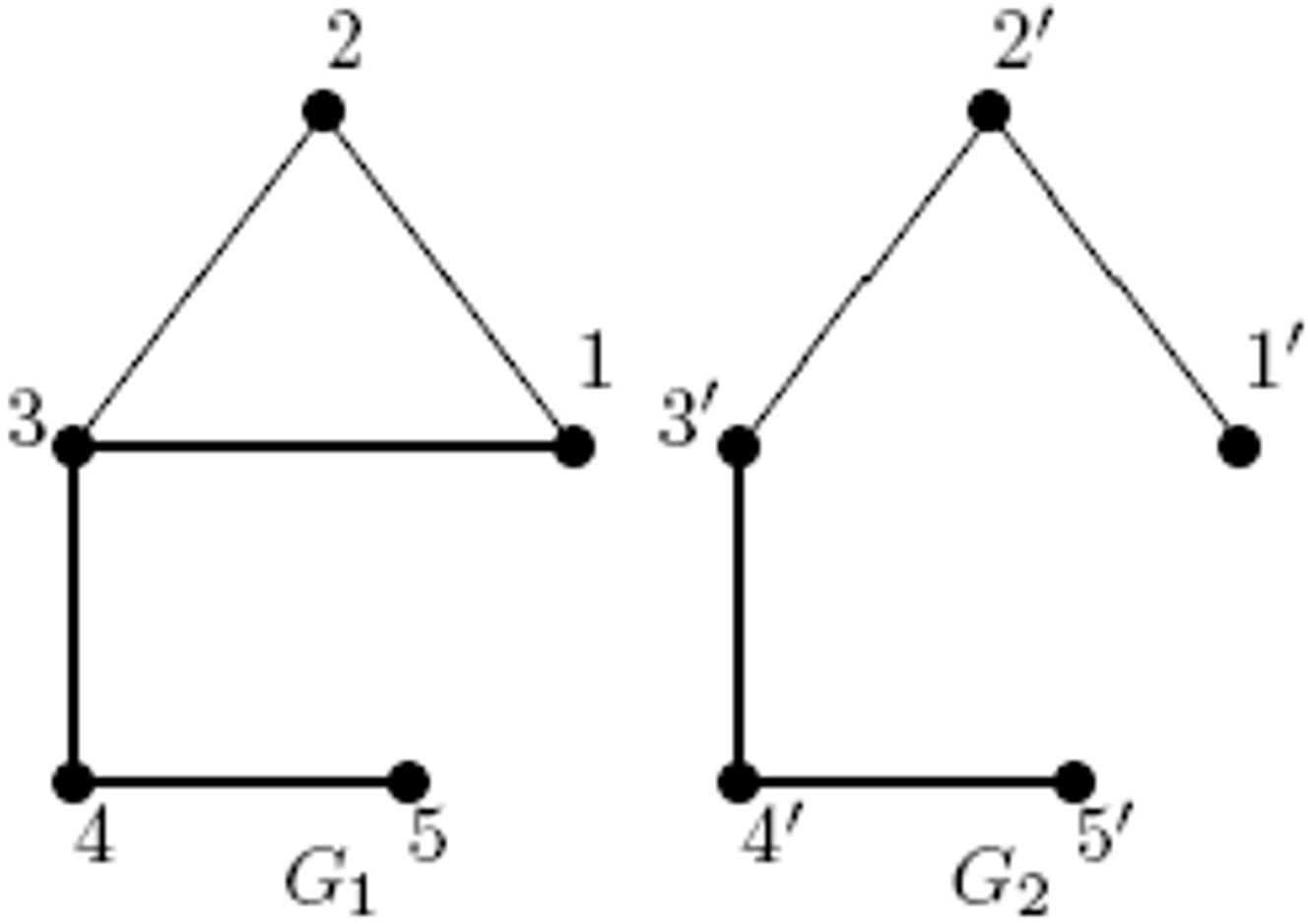

Figure 1
MCIS of two graphs. For G_1 and G_2, the maximum common induced subgraph of them contains four vertices, and the maximum common edge subgraph of them involves five vertices.

of instances of the form (x, k), where x is the problem description and the integer k = 0 is called the *parameter*. The parameterized problem Q is *fixed-parameter tractable* [17] if it can be solved in time $f(k)|x|^{O(1)}$, where f is a recursive function. The class FPT contains all the problems that are fixed-parameter tractable. In this paper, we assume that complexity functions are "nice" with both the domain and range being non-negative integers and the values of the functions and their inverses are easily computed. For two functions f and g, we write f(n) = o(g(n)) if there is a nondecreasing and unbounded function λ such that f(n) = g(n)/λ(n). A function f is *subexponential* if $f(n) = 2^{O(n)}$.

For a problem in the class FPT, research is focused on identifying more efficient, parameterized algorithms. There are many effective techniques to design parameterized algorithm including the methods of "bounded search tree" and "reduction to a problem kernel". Another example is the vertex cover problem.

Definition
Vertex cover problem: given a graph G and an integer k, determine if G has a vertex cover C of k vertices, i.e., a subset C of k vertices in G such that every edge in G has at least one endpoint in C. Here, the parameter is k.

Given a graph of n vertices, there is a parameterized algorithm that can solve the vertex cover problem in time $O(kn + 1.286^k)$ [24].

Accompanying the work on designing efficient and practical parameterized algorithms, a theory of parameter intractability has previously been developed [17]. In parameterized complexity, to classify fixed-parameter intractable problems, a hierarchy of classes (the W-hierar-

chy $\cup_{t=0} W[t]$, where $W[t] \subseteq W[t+1]$ for all $t = 0$) have been introduced in which the 0-th level W [0] is the class *FPT*. The hardness and completeness have been defined for each level W [i] of the W-hierarchy for i = 1, and a large number of W [i]-hard parameterized problems have been identified [17]. For example, the clique problem is W[1]-hard.

Definition

Clique problem: given a graph G and an integer k, determine if G has a clique C of k vertices, i.e., a subset C of k vertices in G such that there is an edge in G between any two of these k vertices, i.e., the k vertices induce a complete subgraph of G. Here the parameter is k.

The clique problem can be solved in time $O(n^k)$, based on the enumeration of all the vertex subsets of size k for a given graph with n vertices.

It has become commonly accepted that no W[1]-hard (and W [i]-hard, i > 1) problem can be solved in time $f(k)n^{O(1)}$ for any function f (i.e., W[1] ? FPT). W[1]-hardness has served as the hypothesis for fixed-parameter intractability. An example is a recent result by Papadimitriou and Yannakakis [25], showing that the database query evaluation problem is W[1]-hard. This provides strong evidence that the problem cannot be solved by an algorithm whose running time is of the form $f(k)n^{O(1)}$, thus excluding the possibility of a practical algorithm for the problem even if the parameter k (the size of the query) is small as in most practical cases.

Based on the W[1]-hardness of the clique algorithm, computational intractability of problems in bioinformatics has been derived [26-31], the author point out that "Unless an unlikely collapse in the parameterized hierarchy occurs, the results proved in [31] that the problems longest common subsequence and shortest common supersequence are W[1]-hard rule 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^{vk} are not deemed impossible by our results."

Recent investigation has derived stronger computational lower bounds for well-known NP-hard parameterized problems [32,33]. 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 bioinformatics problems have been derived [34]. For example, our results for the problem's longest common subsequence and shortest common supersequence have strengthened the results in [31] significantly and advanced the understanding on the complexity of the problems. We show that it is actually unlikely that the problems can be solved in time $n^{\gamma(k)}$ for *any* sublinear function $\gamma(k)$ and the known dynamic programming algorithms of running time $O(n^k)$ for the problems are actually asymptotically optimal.

In the following section, we derive the lower bound for exact algorithms of the maximum common subgraph problem.

Lower Bound for Maximum Common Subgraph Problem

The formal parameterized version of the maximum common subgraph problem is described above; we choose the number of vertices in the common subgraph as the parameter. Based on the reduction from the parameterized clique problem to the parameterized common subgraph problem, we derive the hardness result of the parameterized common subgraph problem.

An NP optimization problem Q is a four-tuple (I_Q, S_Q, f_Q, opt_Q) [19], 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); $S_Q(x) = \{y: |y| = p(|x|) \text{ and } \pi(x, y)\}$;

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 NP optimization problem Q can be parameterized in a natural way as follows [35,32]:

Definition

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

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

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

We now provide the definitions of the maximum common subgraph problem and the parameterized common subgraph problem.

Definition
Maximum common subgraph problem:

Input: two graphs $G_1 = (V_1, E_2)$ and $G_2 = (V_2, E_2)$.

Output: the maximum common vertex-induced subgraph of the two graphs G_1 and G_2.

Definition
Parameterized common subgraph problem:

Input: two graphs $G_1 = (V_1, E_2)$ and $G_2 = (V_2, E_2)$, and a positive integer k;

Parameter: k;

Output: "Yes", if there is a common vertex-induced subgraph of k vertices, i.e., a common subgraph of size k of the two graphs G_1 and G_2. Otherwise, output "No".

Lemma 1
The parameterized common subgraph problem is W[1]-hard.

Proof: We will give an FPT-reduction from clique to the parameterized common subgraph problem as follows.

Given an instance (G, k) of the clique problem, where the graph G has n vertices and k is a positive integer, we construct an instance of the parameterized common subgraph problem as follows: let G_1 be the graph G, and G_2 a complete graph of k vertices. The problem can therefore be stated as "Is a common vertex-induced subgraph of k vertices for the graphs G_1 and G_2?"

We can verify that the graph G has a clique of size k if and only if the graphs G_1 and G_2 have a common subgraph of k vertices. Since the reduction may be finished in polynomial time O(nk), the reduction is an FPT-reduction from clique to parameterized common subgraph problem.

To prove our main result, we will use the definition of linear FPT-reduction and $W_1[1]$-hard [36]:

Definition
A parameterized problem Q is *linear FPT-reducible*, or more precisely, *FPT_l-reducible*, to a parameterized problem Q' if there exist a function f and an algorithm A of running time $f(k)n^{O(1)}$ that, on each (k, n)-instance x of Q, produces a (k', n')-instance x' of Q', where $k' = O(k)$, $n' = n^{O(1)}$, and x is a yes-instance of Q if and only if x' is a yes-instance of Q'.

Linear FPT-reduction has the transitivity property [36,34]. The transitivity of the FPT_l-reduction is proved in the following lemma:

Lemma 2
Let Q_1, Q_2 and Q_3 be three parameterized problems. If Q_1 is FPT_l-reducible to Q_2, and Q_2 is FPT_l-reducible to Q_3, then Q_1 is FPT_l-reducible to Q_3.

Proof: If Q_1 is FPT_l-reducible to Q_2, then there exists a function f_1 and an algorithm A_1 of running time $f_1(k_1)n_1^{o(k1)}m_1^{O(1)}$, such that for each (k_1, n_1, m_1)-instance x_1 of Q_1, the algorithm A_1 produces a (k_2, n_2, m_2)-instance x_2 of Q_2, where $n_2 = n_1^{O(1)}$, $m_2 = m_1^{O(1)}$, and $k_2 = c_1k_1$, where c_1 is a constant.

If Q_2 is FPT_l-reducible to Q_3, then there exists a function f_2 and an algorithm A_2 of running time $f_2(k_2)n_2^{O(k2)}m_2^{O(1)}$, such that on each (k_2, n_2, m_2)-instance x_2 of Q_2, the algorithm A_2 produces a (k_3, n_3, m_3)-instance x_3 of Q_3, where $k_3 = O(k_2)$, $n_3 = n_2^{O(1)}$, $m_3 = m_2^{O(1)}$.

We now have an algorithm A that reduces Q_1 to Q_3, as follows: For a given (k_1, n_1, m_1)-instance x_1 of Q_1, A first calls the algorithm A_1 on x_1 to construct a (k_2, n_2, m_2)-instance x_2 of Q_2, where $k_2 = c_1k_1$, $n_2 = n_1^{O(1)}$, and $m_2 = m_1^{O(1)}$. Then A calls the algorithm A_2 on x_2 to construct a (k_3, n_3, m_3)-instance x_3 of Q_3. It is therefore obvious that x_3 is a yes-instance of Q_3 if and only if x_1 is a yes-instance of Q_1. Moreover, from $k_2 = c_1k_1$ and $k_3 = O(k_2)$, we have $k_3 = O(k_1)$, and from $n_2 = n_1^{O(1)}$, $m_2 = m_1^{O(1)}$, $n_3 = n_2^{O(1)}$, $m_3 = m_2^{O(1)}$, we get $n_3 = n_1^{O(1)}$ and $m_3 = m_1^{O(1)}$. Finally, since the invocation of algorithm A_1 on x_1 takes time $f_1(k_1)n_1^{o(k1)}m_1^{O(1)}$, the invocation of algorithm A_2 on x_2 takes time $f_2(k_2)n_2^{O(k2)}m_2^{O(1)}$, $k_2 = c_1k_1$, $n_2 = n_1^{O(1)}$, and $m_2 = m_1^{O(1)}$, we conclude that the running time of algorithm A is bounded by $f_1(k_1)n_1^{O(k1)}m_1^{O(1)}$, where $f(k_1) = f_1(k_1) + f_2(c_1k_1)$. By definition, A is an FPT_l-reduction from Q_1 to Q_3; i.e., Q_1 is FPT_l-reducible to Q_3.

Definition
A parameterized problem Q is W[1]-hard under the FPT_l-reduction, or more precisely $W_l[1]$-hard, if the *Weighted antimonotone CNF 2SAT* (abbreviated wcnf -2sat$^-$) problem is FPT_l-reducible to Q.

In particular, it has been shown [32,33] that the clique problem is $W_l[1]$-hard.

Lemma 3

(From theorem 5.2 of [33]) Unless all SNP problems are solvable in subexponential time, no $W_l[1]$-hard problem can be solved in time $f(k)n^{O(k)}$ for any recursive function f.

Note Papadimitriou and Yannakakis [30] have introduced the class SNP which contains many well-known NP-hard problems. Some of these problems have been the major targets in the study of exact algorithms, but have so far resisted all efforts for the development of subexponential time algorithms to solve them. Thus, it has been commonly agreed that it is unlikely that all SNP problems are solvable in subexponential time. 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[1,37] to FPT, which is also considered as an unlikely collapse in parameterized computation.

Lemma 4

The parameterized common subgraph problem is $W_l[1]$-hard.

Proof: Referring to the proof of Lemma 1, the reduction from a clique to a parameterized common subgragh problem is a linear FPT-reduction.

Based on the transitivity property of the linear FPT-reduction of Lemma 2, and the fact that the clique problem is $W_l[1]$-hard, the parameterized common subgraph problem could not be solved in time $f(k)n^{O(k)}$, where k is the number of vertices in the common subgraph and f is any recursive function, unless some unlikely collapse (Mini[1] = FPT) occurs in parameterized computation.

From Lemma 4 and Proposition 3, we have the following theorem:

Theorem

Given two graphs G_1 and G_2 with each graph having n vertices, there is no algorithm of time $f(k)n^{O(k)}$ for the parameterized common subgraph problem, where k is the number of vertices in the common subgraph and f is any recursive function, unless some unlikely collapse (Mini[1] = FPT) occurs in parameterized computation.

In consideration of the upper-bound result, we now show that our lower-bound result for the maximum common subgraph problem presented here is asymptotically tight.

Upper Bound – Clique Based Approaches

The following approach for the maximum common subgraph problem is based on the reduction [15,1] from a maximum common subgraph problem to the maximum clique problem.

From two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, a new graph $G = (V, E)$ is derived as follows: Let $V = V_1 \times V_2$ and call V a set of pairs. Call two pairs $\langle u_1, u_2 \rangle$ and $\langle v_1, v_2 \rangle$ compatible if $u_1 \neq v_1$ and $u_2 \neq v_2$ and if they preserve the edge relation, that is, there is an edge between u_1 and v_1 if and only if there is an edge between u_2 and v_2. Let E be the set of compatible edges. A k-clique in the new graph G can be interpreted as a matching between two induced k-node subgraphs. The two subgraphs are isomorphic since the compatible pairs preserve the edge relations. The new graph G is called the modular product graph of the two graphs G_1 and G_2.

We suppose $n = |V_1| = |V_2|$ (The analysis for the case when $|V_1| ? |V_2|$, is similar, and thus is omitted). From the construction of G, we have $|V| = n^2$. By a close observation of the new graph G, we can see that G is indeed an n-partite graph, where the vertices are partitioned into n disjoint partitions with each partition having n vertices.

We may use a matrix to denote the n^2 vertices of the n-partite graph with n vertices in each partition.

$v_{\{1,1\}}, v_{\{1,2\}}, \ldots, v_{\{1,n\}}$

$v_{\{2,1\}}, v_{\{2,2\}}, \ldots, v_{\{2,n\}}$

... ...

$v_{\{n,1\}}, v_{\{n,1\}}, \ldots, v_{\{n,n\}}$

The n vertices of the first row $v_{\{1,i\}}$, $1 = i = n$, belong to partition one of the n-partite graph. The n vertices of the second row $v_{\{2,i\}}$, $1 = i = n$, belong to partition two and so on.

There is no edge between any two vertices within the same partition. Edges only appear between two vertices that are in two different partitions. So, at most one vertex from each partition (of the n vertices) could be in a clique of the graph. Therefore, to find a clique of size k, there will be n^k possible ways for choosing the clique vertices. For each possible way, the algorithm needs $O(k^2)$ time to check if it constructs a clique of size k. Therefore, this gives an algorithm of time $O(n^k k^2)$ for the maximum common subgraph problem. We call this algorithm ALG-COMMON SUBGRAPH for the convenience of the following discussion.

This problem – when the maximum clique size k is equal to n – has been studied by Sze *et al* [38]:

Definition
Given an n-partite graph G with n vertices in each part, the n-CLIQUE$_{np}$ problem finds an n-clique in the graph G.

For this problem, they developed a fast and exact divide-and-conquer approach. The basic idea of this novel approach is to subdivide the given n-partite graph into several n_0-partite subgraphs with $n_0 < n$ and solve each smaller subproblem independently using a branch-and-bound approach as long as the number of cliques of size n_0 in each subproblem is not too high. The reader is referred to [38] for the details of this divide-and-conquer approach. However, their approach in the worst case still has the same upper bound.

Given this $O(n^k k^2)$-time algorithm for the maximum common subgraph problem, the lower bound result of our Theorem is asymptotically tight.

When the number of vertices in the common subgraph k is not very far away from the value of n, we define k = n – c, where c is a constant. We illustrate the basic idea for c = 1 as follows [39]: Suppose the n-partite graph G has a clique C of size k-1. We add one more vertex to each of the n partitions. And we also add edges from this vertex to any vertices (except the newly added vertices) that are not in the same partition. Now we get a new graph G'. G' is an n-partite graph with n + 1 vertices in each partition. The new graph G' has a clique C' of size n if and only if the original n-partite graph G has a clique of size (n-1). The vertices of this clique C' include the vertices of the original clique C and one newly added vertex.

For the newly constructed graph G', we can now apply the algorithm ALG-COMMON SUBGRAPH without any change. And we need time $O((n+1)^n n^2)$. After we find the clique C', we just remove the newly added vertex and return the other vertices of C'.

Similarly, if the n-partite graph G has a clique of size k – c, where c is a positive integer constant, we can find the clique by adding c new vertices and associated edges as described above and then applying the algorithm ALG-COMMON SUBGRAPH which runs in time $O((n+c)^n n^2)$.

This simple idea of dealing with cliques of a size less than n is useful since it makes the algorithm ALG-COMMON SUBGRAPH work uniformly for finding cliques of different sizes on n-partite graphs. In the following, we give the following algorithm for finding cliques of size k – c.

Algorithm for (K-C)-CLIQUE
INPUT: an n-partite graph G, with n vertices in each partition, and a small constant c, where c is a positive integer;

OUTPUT: a clique of size no less than k – c;

Step 1: For i = 0 to c do

• Step 1.1: Construct a new graph G_1, by adding i new vertices to each partition of the graph G and adding edges from each of the new vertices to any vertices (except the newly added vertices) that are not in the same partition.

• Step 1.2: Apply the algorithm ALG-COMMON SUBGRAPH on the graph G_1.

• Step 1.3: If a clique C_1 is found, then return "a clique C of size k – i has been found" (C is constructed by removing all the newly-added vertices from the clique C_1).

• Endfor

Step 2: Return "no clique has been found".

We now propose two approaches for the maximum common subgraph problem which are based on the relationship between the vertex cover problem and the clique problem:

Algorithm 1: ALG-APPROX-CLIQUE
INPUT: an n-partite graph G, with n vertices in each partition, and a small constant c, where c is a positive integer;

OUTPUT: a clique for the graph G.

Step 1. Compute the complement graph G' of the modular product graph G = (V, E) of graph G_1 and G_2;

Step 2. Apply the approximation algorithm for the vertex cover problem to get a vertex cover C;

Step 3. Return V – C as the clique vertex set.

ALG-APPROX-CLIQUE gives an approximate solution for the maximum common subgraph problem in polynomial time. This approach uses the following approximation algorithm for the vertex cover problem with an approximation ratio 2 in [40]:

ALG-APPROX-VERTEX COVER
INPUT: a graph G = (V, E);

OUTPUT: a vertex cover C of approximation ratio 2 for the graph G.

Step 1. $C \leftarrow \Phi$;

Step 2. $E' \leftarrow E(G)$;

Step 3. While $E' \neq \Phi$

• Step 3.1. Let (u, v) be an arbitrary edge of E';

• Step 3.2. $C = C \cup \{u, v\}$;

• Step 3.3. Remove from E' every edge incident on either u or v;

Step 4. Return C as the vertex cover set.

In this algorithm, ALG-APPROX-VERTEX COVER selects an edge from the set of edges of the graph G = (V, E) and adds it to C. Repeating this procedure for $(u, v) \in E(G)$ and deleting edges from E' that are covered by u or v results in a running time of O(V+E).

Algorithm 2: ALG-EXACT-MAXCLIQUE

INPUT: an n-partite graph G, with n vertices in each partition, and a small constant c;

OUTPUT: a clique for the graph G.

Step 1. Compute the complement graph G' of the modular product graph G = (V, E) of graph G_1 and G_2;

Step 2. Apply the parameterized exact algorithm for the Vertex Cover problem on G' and compute the minimum vertex cover C_0.

Step 3. Return the maximum clique with the vertex set V – C_0.

Alternatively, ALG-EXACT-MAXCLIQUE could apply in Step 2 the current best algorithm for vertex cover [24] which is of time $O(kn + 1.286^k)$. By running the vertex cover algorithm for at most n times, we produce the minimum vertex cover of the product graph G.

Results

In this paper we investigated the lower-bound result for the maximum common subgraph problem. We proved that it is unlikely that there is an algorithm of time $f(k)n^{O(k)}$ for the problem, where k is the number of vertices in the common subgraph and f is any recursive function. We then presented the upper bound of algorithms which solve this problem: $O(n^k k^2)$ time where k is the number of vertices in the common subgraph. In consideration of the upper-bound result, we point out that our lower-bound result for the maximum common subgraph problem is asymptotically tight.

Conclusion

Parameterized computation is a viable approach with great potential for investigating many applications within bioinformatics, such as the maximum common subgraph problem studied in this paper. With an improved hardness result and the proposed approaches in this paper, future research can be focused on further exploration of efficient approaches for different variants of this problem within the constraints imposed by real applications.

Authors' contributions

XH carried out the study on the lower bound and approaches for the maximum common subgraph problem and helped to provide background information on parameterized computation theory. JL and SFJ participated in the design and expression of the algorithms for the maximum common subgraph problem. All authors have read and approved the final manuscript.

Acknowledgements

This publication was made possible in part by NIH Grant #P20 RR-16460 from the IDeA Networks of Biomedical Research Excellence (INBRE) Program of the National Center for Research Resources.

This article has been published as part of *BMC Bioinformatics* Volume 7, Supplement 4, 2006: Symposium of Computations in Bioinformatics and Bioscience (SCBB06). The full contents of the supplement are available online at http://www.biomedcentral.com/1471-2105/7?issue=S4.

References

1. Raymond JW, Willett P: **Maximum common subgraph isomorphism algorithms for the matching of chemical structures.** *Journal of Computer-aided Molecular Design* 2002, **16:**521-533.
2. Horaud R, Skordas T: **Stereo correspondence through feature grouping and maximal cliques.** *IEEE Trans Pattern Anal Mach Intell* 1989, **11(11):**1168-1180.
3. Shearer K, Bunke H, Venkatesh S: **Video indexing and similarity retrieval by largest common subgraph detection using decision trees.** *No. IDIAP-RR 00–15, Dalle Molle Institute for Perceptual Artificial Intelligence, Martigny, Valais, Switzerland* 2000.
4. Bowie J, Luthy R, Eisenberg D: **A method to identify protein sequences that fold into a known three-dimensional structure.** *Science* 1991, **253:**164-170.
5. Bryant SH, Altschul SF: **Statistics of sequence-structure threading.** *Curr Opin Struct Biol* 1995, **5:**236-244.
6. Xu Y, Xu D, Uberbacher EC: **An efficient computational method for globally optimal threading.** *Journal of Computational Biology* 1998, **5(3):**597-614.
7. Lathrop RH, Rogers RG Jr, Bienkowska J, Bryant BMK, Buturovic LJ, Gaitatzes C, Nambudripad R, White JV, Smith TF: **Analysis and algorithms for protein sequencestructure alignment.** In *Computational Methods in Molecular Biology, Salzberg, Searls* Edited by: *Kasif. Elsevier*; 1998.
8. Xu J, Li M, Kim D, Xu Y: **RAPTOR: optimal protein threading by linear programming.** *J Bioinform Comput Biol* 2003, **1(1):**95-117.
9. Doudna JA: **Structural genomics of RNA.** *Nature Structural Biology* 2000, **7(11 supp):**954-956.
10. Eddy SR: **Computational genomics of non-coding RNA genes.** *Cell* 2002, **109:**137-140.
11. Rivas E, Eddy SR: **Noncoding RNA gene detection using comparative sequence analysis.** *BMC Bioinformatics* 2001, **2:**8.
12. Lowe TM, Eddy SR: **tRNAscan-SE: A Program for improved detection of transfer RNA genes in genomic sequence.** *Nucleic Acids Research* 1997, **25:**955-964.

13. Song Y, Liu C, Huang X, Malmberg R, Xu Y, Cai L: **Efficient parameterized algorithm for biopolymer structure-sequence alignment.** *Proceedings of 5th Workshop on Algorithms in BioInformatics (WABI 2005), Lecture Notes in Bioinformatics* 2005, **3692:**376-388.
14. Gary MR, Johnson DS, Computers and Intractability: *a Guide to the Theory of NP-Completeness WH. Freeman and Co*; 1979.
15. Kann V: **On the approximability of the maximum common subgraph problem.** In *Proc 9th Annual Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science 577 Springer-Verlag*; 1992:377-388.
16. Cheetham J, Dehne F, Rau-Chaplin A, Stege U, Taillon PJ: **Solving large FPT problems on coarse-grained parallel machines.** *JCSS* 2003, **67:**691.
17. Downey R, Fellows M: *Parameterized Complexity Springer*; 1999.
18. Lanctot JK, Li M, Ma B, Wang S, Zhang L: **Distinguishing string selection problems.** *Inf Comput* 2003, **185:**41.
19. Ausiello G, Crescenzi P, Gambosi G, Kann V, Marchetti-Spaccamela A, Protasi M: *Complexity and Approximation, Combinatorial Optimization Problems and Their Approximability Properties New York: Springer-Verlag*; 1999.
20. Deng X, Li G, Li Z, Ma B, Wang L: **A PTAS for distinguishing (sub)string selection.** *LNCS* 2002, **2380:**740.
21. Deng X, Li G, Li Z, Ma B, Wang L: **Genetic design of drugs without side-effects.** *SIAM Journal on Computing* 2003, **32:**1073.
22. Jiang T, Li M: **On the Approximation of shortest common Supersequences and longest Common subsequences.** *SIAM J Comput* 1995, **24:**1122.
23. Li M, Ma B, Wang L: **On the closest string and substring problems.** *Journal of the ACM* 2002, **49:**157.
24. Chen J, Kanj I, Jia W: **Vertex cover: further observations and further improvements.** *Journal of Algorithms* 2001, **41:**280-301.
25. Papadimitriou C, Yannakakis M: **On the complexity of database queries.** *JCSS* 1999, **58:**.
26. Bodlaender HL, Downey RG, Fellows MR, Hallett MT, Wareham HT: **Parameterized complexity analysis in computational biology.** *Comput Appl Biosci* 1995, **11:**49-57.
27. Bodlaender H, Downey R, Fellows M, Wareham M: **The parameterized complexity of sequence alignment and consensus.** *Theoretical Computer Science* 1995, **147:**31.
28. Fellows M, Gramm J, Niedermeier R: **Parameterized intractability of motif search problems.** *LNCS* 2002, **2285:**262.
29. Hallett M: *An Integrated Complexity Analysis of Problems for Computational Biology Ph.D. Thesis, University of Victoria*; 1996.
30. Papadimitriou C, Yannakakis M: **On limited nondeterminism and the complexity of VC dimension.** *JCSS* 1996, **53:**161.
31. Pietrzak K: **On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems.** *JCSS* 2003, **67:**757.
32. Chen J, Chor B, Fellows M, Huang X, Juedes D, Kanj I, Xia G: **Tight lower bounds for parameterized NP-hard problems.** *Proc of the 19th Annual IEEE Conference on Computational Complexity* 2004:150-160.
33. Chen J, Huang X, Kanj I, Xia G: **Linear FPT reductions and computational lower bounds.** *Proc of the 36th ACM Symposium on Theory of Computing* 2004:212-221.
34. Huang X: *Parameterized Complexity and Polynomial-time Approximation Schemes Ph.D. Dissertation, Texas A&M University*; 2004.
35. Cai L, Chen J: **On Fixed-Parameter Tractability and Approximability of NP Optimization Problems.** *J Comput Syst Sci* 1997, **54:**465-474.
36. Chen J, Huang X, Kanj I, Xia G: **W-hardness linear FPT-reductions: structural properties and further applications.** *Proceedings of the Eleventh International Computing and Combinatorics Conference (COCOON 2005), Lecture Notes in Computer Science* 2005, **3595:**975-984.
37. Downey R, Estivill-Castro V, Fellows M, Prieto E, Rosamond F: **Cutting up is hard to do: the parameterized complexity of k-Cut and related Problems.** *Electr Notes Theor Comput Sci* 2003, **78:**.
38. Sze S-H, Lu S, Chen J: **Integrating sample-driven and pattern-driven approaches in motif finding.** *WABI2004* 2004:438-449.
39. Sze S-H: *Lectures notes of Special Topics in Computational Biology, Fall* 2002.
40. Cormen TH, Leiserson CE, Rivest RL, Stein C: *Introduction to Algorithms* 2nd edition. *MIT Press*; 2001.