Parameterized Complexity in Practice
Examples in Bioinformatics

Luke Mathieson  Wagner Costa  Pritha Mahata
Drew Mellor  Pablo Moscato  Elena Prieto  Carlos Riveros

The Centre for Bioinformatics, Biomarker Discovery & Information-Based Medicine

Workshop on Parameterized Complexity and Algorithm Design
30–31 March, 2010
University of Newcastle, Australia
Why is Parameterized Complexity relevant?

- Most interesting problems are formally hard to solve.
- Many are even hard to approximate well.
  - And sometimes approximations just aren’t good enough.
- Other models of computation aren’t usable, and probably don’t help.

So we’re really left with two approaches to deal with the problem of intractability; Exact (Exponential) Algorithms, and Parameterized Complexity
Bioinformatics problems typically involve large inputs (datasets, networks, etc.).

They also involve lots of structure that we can take advantage of. = lots of natural parameters!
Hierarchical Clustering

Given \( n \) elements with an \( n \times n \) distance matrix \( D \) and some objective function the *hierarchical clustering* problem asks for a hierarchical partitioning of the elements such that the partitioning at each level satisfies the objective function. Then the choice of objective function determines the effectiveness of the clustering.
Hierarchical clustering is interesting for explaining relationships for many biological problems and particularly for:

- gene expression clustering (e.g., [Kull08, Qin03, Seal05]),
- phylogenetic clustering (e.g., [Ailon05, Jothi06]).

Of course we don’t want to do this by hand, so we need good algorithms.
Many approaches for hierarchical clustering exist:

- Agglomerative Hierarchical Clustering
- $k$-Means Clustering
- Graph Bipartitioning
  - Maximum Cut
  - Normalized Cut ([Shi00])

Jain et al. review older methods for hierarchical clustering. Most methods have some problems with various datasets (typically they don’t account for intracluster similarity).

Of course most effective methods are also NP-hard and hard to approximate.
A Better(?) Method

(First proposed by Mahata et al. [Mahata06])

**Arithmetic-Harmonic Cut (AH-Cut)**

*Instance:* A graph $G = (V, E)$, two positive integers $k$ and $d$ and a weight function $\omega : E \rightarrow [1, d]$.

*Question:* Is there a partition of $V$ into two sets $B$ and $W$ such that

$$
(\sum_{uv \in E_{BW}} \omega(uv))\left(\sum_{uv \in E \setminus E_{BW}} \frac{1}{\omega(uv)}\right) \geq k?
$$

So the objective function is proportional to the sum of the distances between the two groups, and the sum of the inverses of the distances within the groups.
AH-Cut is:

- NP-complete (as usual),
- APX-hard (so we can’t approximate too well),
- but luckily, fixed-parameter tractable if we take $k + d$ as the parameter.

Even better, it is fixed-parameter tractable using the greedy localisation technique, so a simple greedy algorithm either gives a good answer, or we’re not so far from one.
Needles, Haystacks and Microarrays

The haystack:
Microarrays allow quick generation of large gene expression datasets.

The needle:
How do we search through all this data to find the genes that actually mean something?

- We can model this a graph problem!

This is a specific application of the more general inference problem.
The Combinatorial Approach

We have:

- a vertex for each gene;
- a vertex for each pair of samples (patients, cell lines, etc.);
- and an edge between a gene vertex and a sample pair vertex if the expression of the gene discriminates the two samples (i.e. if the two samples are from different classes and their expression for that gene differs).

So then if we can find a subset of the gene vertices so that there is an edge between at least one of these and every sample pair vertex, then this subset will explain the differences in the examples.

If we have even only 100 samples, we get 10,000 sample pair vertices, so we can’t really do this by hand, even if we ignore that we likely have several thousand gene vertices too.
(A Graph Version of) **Hitting Set**

**Hitting Set**

*Instance:* A bipartite graph \( G = (V_1 \cup V_2, E) \), a positive integer \( k \).

*Question:* Is there a subset \( V' \subset V_1 \) of size at most \( k \) such that for every vertex \( v \in V_2 \) there is a vertex \( u \in V' \) where \( uv \in E \)?

This is equivalent to the \( k\)-Feature Set problem (amongst others) [Cotta03, Cotta07].
Unfortunately **Hitting Set** is $W[2]$-complete [Paz81], so we can’t get a fixed-parameter tractable algorithm unless the $W$-hierarchy collapses, which seems unlikely.

Fortunately there’s two ways we can attempt to overcome this difficulty:

- restrict the problem;
- restrict the input.
**d-Hitting Set**

*Instance: A bipartite graph $G = (V_1 \cup V_2, E)$ where $d(v) \leq d$ for all $v \in V_2$, a positive integer $k$.*

*Question: Is there a subset $V' \subset V_1$ of size at most $k$ such that for every vertex $v \in V_2$ there is a vertex $u \in V'$ where $uv \in E$?*

This version is fixed-parameter tractable with parameter $k$ where $d$ is a constant [Abu-Khzam09].
A Tractable Version of **Hitting Set**

We can even cover more ground and still retain fixed-parameter tractability [Mellor10]:

\[(m, d)\text{-Hitting Set}\]

**Instance:** A bipartite graph \(G = (S \cup C, E)\) where for all \(c \in C\) we have \(d(c) \leq d\), a hitting function \(\eta : C \to [0, m]\) and an integer \(k\).

**Question:** Is there a set \(S' \subseteq S\) of size at most \(k\) such that for every \(c \in C\) we have \(|c \cap S'| \geq \eta(c)|\)?
What if the input isn’t so bad?

We can take a less formal approach:
Consider the following safe data reduction rules for the basic Feature Set problem:

1. If we have two gene vertices $u$ and $v$ where $u$’s neighbours are all neighbours of $v$, then we may delete $u$, as $v$ will cover at least as many example pair vertices.

2. If we have two example pair vertices $x$ and $y$ where $x$’s neighbours are all neighbours of $y$, then we may delete $y$, as any explanation for $x$ will explain $y$.

Surprisingly, these two simple rules are extremely effective in practice - it seems the practical inputs all come from some subset of the theoretically possible inputs.
Conclusion

- Large problems need good algorithms, and when things are NP-complete, we need way of coping.
- Many (biological) problems are combinatorial in nature.
- Parameterized Complexity gives practical methods for dealing with intractable problems, and when it doesn’t, good reasons why not.
Thanks!

Questions?


References III


