Optimally Solving Hard Combinatorial Problems in Computational Biology

Falk Hüffner

Institut für Softwaretechnik und Theoretische Informatik, TU Berlin

7 October 2013

Multiple Sequence Alignment

T_1 A_2 C_3 G_4 T_5 A_6

T_1 A_2 G_3 T_4 A_5

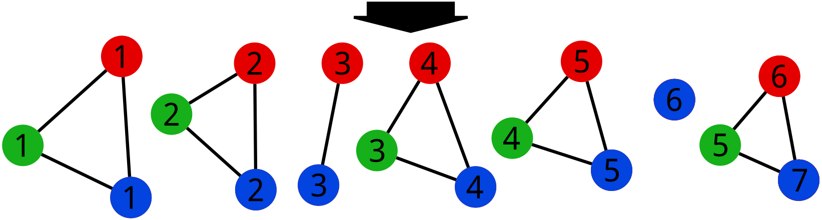
T_1 A_2 C_3 G_4 T_5 G_6 A_7

Multiple Sequence Alignment

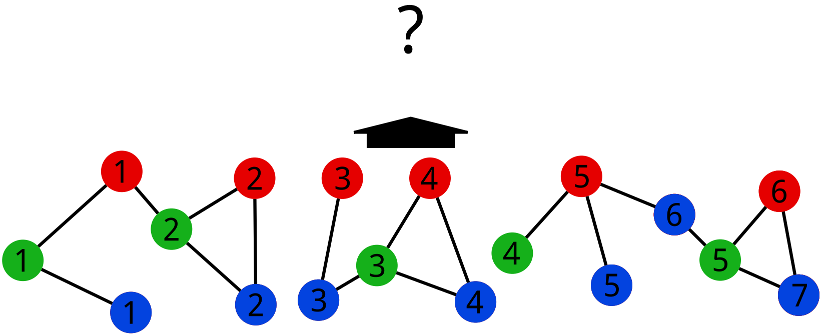
T_1	A_2	C_3	G_4	T_5		A_6
T_1	A_2		G_3	T_4		A_5
T_1	A_2	C_3	G_4	T_5	G_6	A_7

Multiple Sequence Alignment

T_1	A_2	C_3	G_4	T_5		A_6
T_1	A_2		G_3	T_4		A_5
T_1	A_2	C_3	G_4	T_5	G_6	A_7



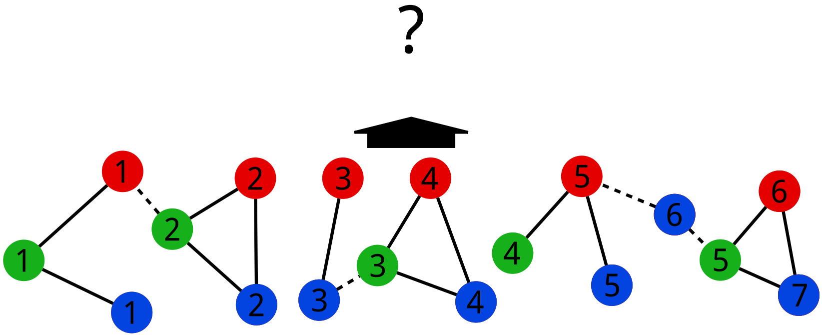
Multiple Sequence Alignment



Idea

Use alignment graph constructed by local alignment to reconstruct global alignment.

Multiple Sequence Alignment

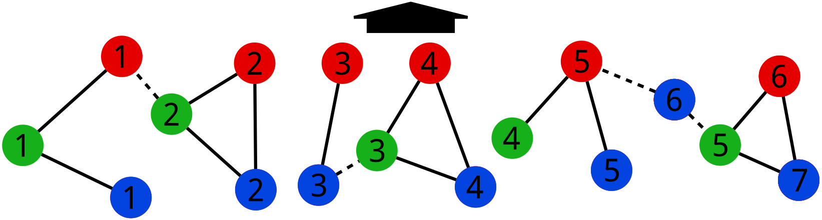


Idea

Use alignment graph constructed by local alignment to reconstruct global alignment.

Multiple Sequence Alignment

T_1	A_2	C_3	G_4	T_5		A_6
T_1	A_2		G_3	T_4		A_5
T_1	A_2	C_3	G_4	T_5	G_6	A_7



Idea

Use alignment graph constructed by local alignment to reconstruct global alignment.

Colorful Components

Part of a Multiple Sequence Alignment pipeline suggested by Corel, Pitschi & Morgenstern (Bioinformatics 2010).

Colorful Components

Part of a Multiple Sequence Alignment pipeline suggested by Corel, Pitschi & Morgenstern (Bioinformatics 2010).

Colorful Components

Instance: An undirected graph $G = (V, E)$ and a coloring of the vertices $\chi : V \to \{1, \ldots, c\}$.
Task: Delete a minimum number of edges such that all connected components are *colorful*, that is, they do not contain two vertices of the same color.

Colorful Components

Part of a Multiple Sequence Alignment pipeline suggested by Corel, Pitschi & Morgenstern (Bioinformatics 2010).

Colorful Components

Instance: An undirected graph $G = (V, E)$ and a coloring of the vertices $\chi : V \to \{1, \ldots, c\}$.
Task: Delete a minimum number of edges such that all connected components are *colorful*, that is, they do not contain two vertices of the same color.

Other application: Orthologs in multiple genomes: From the set of all pairwise homologies, find disjoint orthology sets of genes.
[Zheng, Swenson, Lyons & Sankoff, WABI ’11]

Complexity of Colorful Components

- Colorful Components with two colors can be solved in $O(\sqrt{n}m)$ time by matching techniques.

Complexity of Colorful Components

- Colorful Components with two colors can be solved in $O(\sqrt{n}m)$ time by matching techniques.
- Colorful Components is NP-hard already with three colors.

Complexity of Colorful Components

- Colorful Components with two colors can be solved in $O(\sqrt{n}m)$ time by matching techniques.
- Colorful Components is NP-hard already with three colors.
- Colorful Components can be approximated by a factor of $4\ln(c+1)$.

Exact solutions

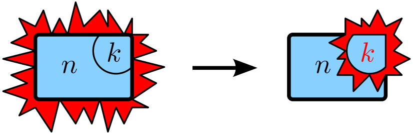
Want to solve COLORFUL COMPONENTS exactly:

- Can interpret solutions within the model;
- Can differentiate between weaknesses of model and weaknesses of algorithm;
- Can judge quality of heuristics;
- Time-limited exact algorithms often give good heuristics.

Fixed-parameter algorithms

Idea

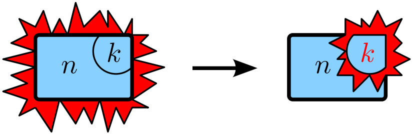
Find an algorithm that gives optimal solutions and thus has exponential running time, but restrict the combinatorial explosion to a *parameter*.



Fixed-parameter algorithms

Idea

Find an algorithm that gives optimal solutions and thus has exponential running time, but restrict the combinatorial explosion to a *parameter*.



Definition

A problem is called fixed-parameter tractable with respect to a parameter k if an instance of size n can be solved in $f(k) \cdot n^{O(1)}$ time for an arbitrary function f.

Fixed-parameter algorithm

Observation

COLORFUL COMPONENTS can be seen as the problem of destroying by edge deletions all bad paths, that is, simple paths between equally colored vertices.

Fixed-parameter algorithm

Observation

Colorful Components can be seen as the problem of destroying by edge deletions all bad paths, that is, simple paths between equally colored vertices.

Observation

Unless the graph is already colorful, we can always find a bad path with at most c edges, where c is the number of colors.

Fixed-parameter algorithm

Observation

COLORFUL COMPONENTS can be seen as the problem of destroying by edge deletions all bad paths, that is, simple paths between equally colored vertices.

Observation

Unless the graph is already colorful, we can always find a bad path with at most c edges, where c is the number of colors.

Theorem

COLORFUL COMPONENTS can be solved in $O(c^k \cdot m)$ time, where k is the number of edge deletions.

Improved fixed-parameter algorithm

Theorem

Colorful Components can be solved in $O((c-1)^k \cdot m)$ time, where k is the number of edge deletions.

Improved fixed-parameter algorithm

Theorem

Colorful Components can be solved in $O((c-1)^k \cdot m)$ time, where k is the number of edge deletions.

Proof.

If there is a degree-3 or higher vertex v, find a bad path with at most $(c-1)$ edges by BFS from v. Otherwise, the instance is easy. □

Limits of fixed-parameter algorithms

Question

How much further can we improve this algorithm?

Limits of fixed-parameter algorithms

Question

How much further can we improve this algorithm?

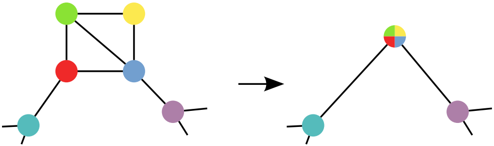
Theorem

Colorful Components with three colors cannot be solved in $2^{o(k)} \cdot n^{O(1)}$ unless the Exponential Time Hypothesis is false.

Data reduction

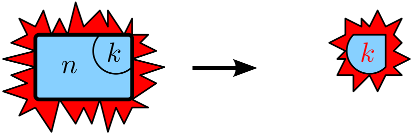
Data reduction

Let $V' \subseteq V$ be a colorful subgraph. If the cut between V' and $V \setminus V'$ is at least as large as the connectivity of V', then merge V' into a single vertex.



Kernelizations

- In classical (one-dimensional) complexity analysis, nothing can be proven about the power of data reduction.
- In parameterized complexity, we have the concept of a *problem kernel*: a data reduction rule that creates an instance whose size depends only on the parameter k, and not on the original input size n anymore.



Data

- We generated one Colorful Components instance for each multiple alignment instance from the BAliBASE 3.0 benchmark.
- We restricted the experiments to the 135 of instances that have at most 10 colors.

Data reduction: Largest connected component

	original			after data red.		
	n	m	c	n	m	c
average	504	921	6.2	354	607	5.3
median	149	232	6	42	58	5

Branching algorithms: running time

	< 1 s	1 s to 10 min	> 10 min
branching branching	70	9	56

Sequence alignment quality

DIALIGN with several methods for solving the COLORFUL COMPONENTS subproblem:

	TC score
min-cut heuristic	53.6 %
exact algorithm	56.6 %

Sequence alignment quality

DIALIGN with several methods for solving the COLORFUL COMPONENTS subproblem:

	TC score
min-cut heuristic	53.6 %
exact algorithm	56.6 %

DIALIGN with the min-cut heuristic is about 10 percentage points worse than current state-of-the-art multiple alignment methods. Hence, an improvement of 3 percentage points is a sizable step towards closing the gap between DIALIGN and these methods.

Integer Linear Programming

An Integer Linear Program (ILP) minimizes a linear function under linear constraints and integrality constraints.

Integer Linear Programming

An Integer Linear Program (ILP) minimizes a linear function under linear constraints and integrality constraints.

Commercial solvers like CPLEX and Gurobi profit from decades of engineering and can often solve real-world instances surprisingly fast.

ILP for Colorful Components

Idea: binary variable e_{uv} that is 1 if u and v are in the same component

ILP for Colorful Components

Idea: binary variable e_{uv} that is 1 if u and v are in the same component

maximize $\sum_{\{u,v\}\in E} w_{uv} e_{uv}$ where

$$w_{uv} = \begin{cases} -\infty & \text{if } \chi(u) = \chi(v), \\ 1 & \text{if } \{u, v\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

ILP for Colorful Components

Idea: binary variable e_{uv} that is 1 if u and v are in the same component

maximize $\sum_{\{u,v\} \in E} w_{uv} e_{uv}$ where

$$w_{uv} = \begin{cases} -\infty & \text{if } \chi(u) = \chi(v), \\ 1 & \text{if } \{u, v\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

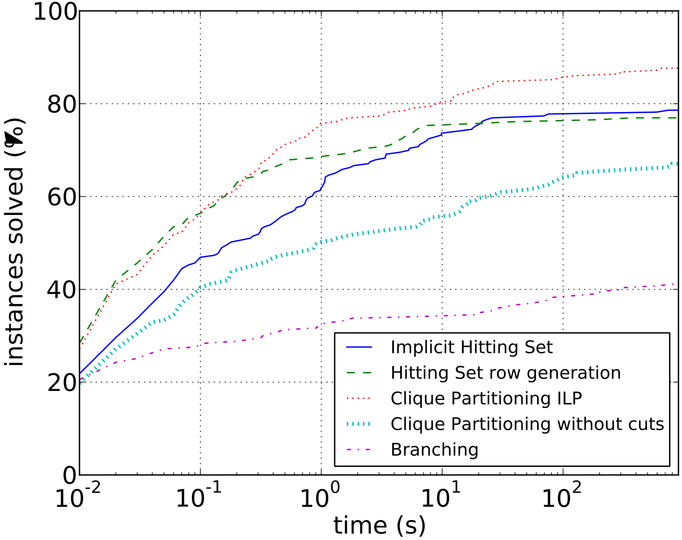
subject to

$$e_{uv} + e_{vw} - e_{uw} \leqslant 1$$

Wikipedia interlanguage links

- 30 most popular languages
- 11,977,500 vertices, 46,695,719 edges
- 2,698,241 connected components, of which 2,472,481 are already colorful
- largest connected component has 1,828 vertices and 14,403 edges
- solved optimally by data reduction + ILP in about 80 minutes
- 618,660 edges deleted, 434,849 inserted.

Random graph model



Graph orientation

- Current technologies like two-hybrid screening can find protein interactions, but cannot decide their direction.
- We can try to reconstruct the directions from gene knockout experiments.

Graph orientation

- Current technologies like two-hybrid screening can find protein interactions, but cannot decide their direction.
- We can try to reconstruct the directions from gene knockout experiments.

Graph Orientation

Instance: An undirected graph $G = (V, E)$ and a set $P \subseteq V \times V$ of source–target pairs.
Task: Find an orientation of each edge in E such that for a maximum number of $(s, t) \in P$ there is a directed path from s to t.

Graph orientation

Data reduction

Join the vertices of a cycle into a single vertex.

Observation

We can assume w. l. o. g. that the input is a tree.

Graph orientation

Data reduction

Join the vertices of a cycle into a single vertex.

Observation

We can assume w. l. o. g. that the input is a tree.

Theorem (Medvedovsky, Bafna, Zwick & Sharan, WABI ’08)

TREE ORIENTATION is NP-hard, even if the tree has diameter two or maximum degree three.

Graph orientation

Data reduction

Join the vertices of a cycle into a single vertex.

Observation

We can assume w. l. o. g. that the input is a tree.

Theorem (Medvedovsky, Bafna, Zwick & Sharan, WABI '08)

Tree Orientation is NP-hard, even if the tree has diameter two or maximum degree three.

Theorem

Tree Orientation can be reduced to Vertex Cover on the conflict graph.

Parameters

p: number of pairs: $2^p \cdot n^{O(1)}$
k: number of unsatisfied pairs: $1.38^k \cdot n^{O(1)}$
m_v: max. number of paths over a tree vertex: $2^{m_v} \cdot n^{O(1)}$
q_v: max. number of cross paths over a tree vertex: $2^{q_v} \cdot n^{O(1)}$
m_e: max. number of paths over an edge: NP-hard for $m_e \geqslant 3$

Parameters

p: number of pairs: $2^p \cdot n^{O(1)}$
k: number of unsatisfied pairs: $1.38^k \cdot n^{O(1)}$
m_v: max. number of paths over a tree vertex: $2^{m_v} \cdot n^{O(1)}$
q_v: max. number of cross paths over a tree vertex: $2^{q_v} \cdot n^{O(1)}$
m_e: max. number of paths over an edge: NP-hard for $m_e \geqslant 3$

n	p	k	m_v	q_v	m_e
799	2014	17	2014	3	59
796	2443	46	2443	35	275
638	2311	68	2310	151	208
441	787	75	785	88	45
299	477	110	411	75	165
192	167	32	161	24	86
114	27	2	26	2	21

Experiments

Data reduction for Vertex Cover

Take the neighbor of a degree-1 vertex into the cover.

Running time:

Branching	0.13s
ILP	0.02s

Conclusions

- Fixed-parameter algorithms can often solve hard problems optimally and have useful worst-case bounds;
- Data reduction can substantially speed up algorithms for hard problems and should always be used, whether using exact or heuristic approaches;
- Integer Linear Programming often yields a simple way to solve hard problems fast.