Combinatorial Optimization and Integer Linear Programming

Combinatorial Optimization: Introduction

Many problems arising in practical applications have a special, *discrete* and *finite*, nature:

Definition. (Linear Combinatorial Optimization Problem)
Given

- a finite set E (the ground set),
- a subset $\mathcal{F} \subseteq 2^E$ (the set of feasible solutions),
- a cost function $c : E \to \mathbb{R}$,

find a set $F^* \in \mathcal{F}$ such that

$$c(F^*) := \sum_{e \in F^*} c(e)$$

is maximal or minimal.

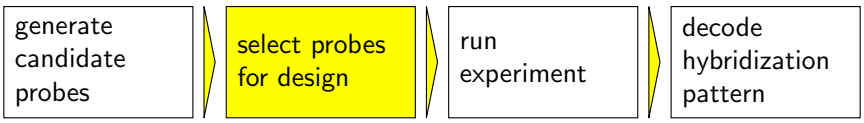
Examples: Shortest Path, Traveling Salesman, and many many more...
Just in bioinformatics: Alignments, Threading, Clone-Probe Mapping, Probe Selection, *De Novo* Peptide Sequencing, Side-Chain Placement, Maximum-weight Connected Subgraph in PPI Networks, Genome Rearrangements, Cluster Editing, Finding Regulatory Modules, Finding Approximate Gene Clusters, Haplotyping, and many more...

Combinatorial Optimization: Introduction (2)

Example. Optimal Microarray Probe Selection

Experimental setup (group testing):

- Goal: determine presence of *targets* in sample
- *probes* hybridize with *targets* $\to$ hybridization pattern



Selection phase:

- *unique* probes are easy to decode but difficult to find (similarities, errors, add. constraints, ...)
- $\to$ consider *non-unique probes*
- Task: choose *few* probes that still allow to infer which targets are in the sample

Combinatorial Optimization: Introduction (3)

Example hybridization matrix $(H)_{ij}$:

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
t_1	1	1	1	0	1	1	0	0	0
t_2	1	0	1	1	0	0	1	1	0
t_3	0	1	1	1	0	1	1	0	1
t_4	0	1	0	0	1	0	1	1	1

Assume: no errors, only *one* target present in sample

Combinatorial Optimization: Introduction (4)

Example hybridization matrix $(H)_{ij}$:

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
t_1	1	1	1	0	1	1	0	0	0
t_2	1	0	1	1	0	0	1	1	0
t_3	0	1	1	1	0	1	1	0	1
t_4	0	1	0	0	1	0	1	1	1

Assume: no errors, only *one* target present in sample

Combinatorial Optimization: Introduction (5)

Example hybridization matrix $(H)_{ij}$:

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
t_1	1	1	1	0	1	1	0	0	0
t_2	1	0	1	1	0	0	1	1	0
t_3	0	1	1	1	0	1	1	0	1
t_4	0	1	0	0	1	0	1	1	1

Assume: no errors, *two* targets present, e.g., t_2 and t_3

Combinatorial Optimization: Introduction (6)

Example hybridization matrix $(H)_{ij}$:

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
t_1	1	1	1	0	1	1	0	0	0
t_2	1	0	1	1	0	0	1	1	0
t_3	0	1	1	1	0	1	1	0	1
t_4	0	1	0	0	1	0	1	1	1

Assume: no errors, *two* targets present, e.g., t_2 and t_3

Details and solution approach can be found in

- G. W. Klau, S. Rahmann, A. Schliep, M. Vingron, K. Reinert: Integer Linear Programming Approaches for Non-Unique Probe Selection. Discrete Applied Mathematics, volume 155, Issues 6-7, pp. 840-856, 2007.

Combinatorial Optimization: Introduction (7)

We want to solve the following problem.

Definition. Probe Selection Problem (PSP)

- Given an incidence matrix H, $d \in \mathbb{N}$, and $c \in \mathbb{N}$,
- find the smallest subset $D \subseteq N$, such that
  - all *targets* are covered by at least d probes
  - all different subsets of targets S and T up to cardinality c are d-separable with respect to D

Observation. PSP is a combinatorial optimization problem, because

- ground set = candidate probes, i.e., $E := \{1, 2, \ldots, n\}$.
- feasible solutions = feasible designs, i.e.,

$$\mathcal{F} := \{D \in 2^E \mid D \text{ satisfies coverage and separation constraints}\}$$

- all costs $c(e) := 1$.

Combinatorial Optimization: Introduction (8)

More examples. What about

$$\min\{3x^2 + 2 \mid x \in \mathbb{R}\} \ ?$$

Or

$$\begin{aligned} \max \quad & 2x_1 + 3x_2 \\ \text{s.t.} \quad & x_1 + 2x_2 \leq 3 \\ & 3x_1 - x_2 \leq 5 \\ & x_1, x_2 \in \mathbb{N} \ ? \end{aligned}$$

Interesting combinatorial problems have an exponential number of feasible solutions. [Otherwise, a straightforward polynomial-time algorithm finds optimal solutions.]

Combinatorial optimization: find solutions faster than by complete enumeration.

Combinatorial Optimization (9)

Now, given a combinatorial optimization problem $C = (E, \mathcal{F}, c)$, we define, for each feasible solution $F \in \mathcal{F}$, its *characteristic vector* $\chi^F \in \{0,1\}^E$ as

$$\chi^F_e := \begin{cases} 1 & e \in F \\ 0 & \text{otherwise} \end{cases} .$$

Then, assuming the objective is to maximize, C can be seen as maximizing over a polytope, i. e.,

$$\max\{c^T x \mid x \in \operatorname{conv}\{\chi^F \in \{0,1\}^E \mid F \in \mathcal{F}\} \ .$$

Why *polytope*?

Theorem. (Minkowski 1896, Weyl 1935)
Each polytope $P = \{x \in \mathbb{R}^n \mid Ax \leq b, l \leq x \leq u\}$ can be written as

$$P = \operatorname{conv}(V)$$

where V is a finite subset of $\mathbb{R}^n$ *and vice versa*.

Combinatorial Optimization (10)

It is possible to switch between these descriptions as $\mathcal{H}$*-polytope* (halfspaces) and $\mathcal{V}$*-polytope* (vertices) with the *Fourier-Motzkin elimination* method.

Example.

Consider the $\mathcal{V}$-polytope defined by

$$P = \text{conv}\left\{\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\1\end{pmatrix}\right\}$$

End (Example).

So, we can just compute the $\mathcal{H}$-polytope $\{x \in \mathbb{R}^n \mid Ax \leq b\}$ for C and optimize over it using, e. g., the Simplex method?

Unfortunately, it is not so easy:

- In general, we cannot find A and b in polynomial time.
- The size of A and b might be exponential.
- The coefficients in A and b can be exponentially large.

Combinatorial Optimization (11)

A little bit of light... often, finding an *integer linear programming (ILP) formulation* is easier:

$$\max\{c'^T x' \mid A'x' \leq b, x' \in \mathbb{Z}\} \ .$$

But: solving LPs is easy, solving ILPs is not!

I will now demonstrate an very useful software package, PORTA, on a simple example that we will encounter later again.

I will demonstrate:

- How to formulate the ILP (and find out that our description in halfspaces is not tight (using `soplex`)).
- How to compute the convex hull of *all* feasible solutions (using `traf`)
- How to enumerate *all* integer points lying in a $\mathcal{H}$-polytope (using `vint`)

Alignments using Combinatorial Optimization

Sources for the following lectures:

- Althaus, E., Caprara, A., Lenhof, H.-P., Reinert, K.: Multiple Sequence alignment with arbitrary gap costs:Computing an optimal solution using polyhedral combinatorics. Proceedings of the 1st European Conference on Computational Biology (ECCB 2002), pages 4-16, 2002
- Althaus, E., Caprara, A., Lenhof, H.-P., Reinert, K.: A Branch-and-Cut Algorithm for Multiple Sequence alignment: Mathematical Programming 2005
- J. D. Kececioglu, H.-P. Lenhof, K. Mehlhorn, P. Mutzel, K. Reinert, M. Vingron: A Polyhedral Approach to Sequence Alignment Problems. Discrete Applied Mathematics, volume 104, pages 143-186, 2000.

Motivation

Multiple alignment and structural multiple alignment are hard combinatorial problems. If one wants to solve them exactly, the usual way is to use dynamic programming combined with clever bounding techniques.

Nevertheless, dynamic programming has clear limitations. If one considers for example the generalization of the pairwise dynamic programming algorithm for multiple alignment, we have to consider $O(2^k)$ choices in *each* step of the algorithm, *and* we have to store intermediate results. This is clearly only feasible for $k \approx 10$ sequences.

Similar statements hold for structural alignments.

Motivation (2)

Why bother? Why do we want to compute *exact* solutions to these problems? Are heuristics not sufficient, especially considering that all scoring schemes only approximate the biological truth?

The answer is twofold:

1. In order to know how well a scoring scheme models the biological truth, we need exact results to compare them to experimentally derived findings. For example we can use the database *BaliBase*, which is based on the alignment of structural protein units to evaluate the performance of a given scoring scheme.
2. If we know that a scoring scheme approximates the problem at hand very well, then it might be worthwile to compute exact solutions if the problem size allows.

Motivation (3)

In the following lectures we introduce a set of different techniques to solve these problems to optimality. These techniques are based on the formulation of the problems as ILPs (*Integer Linear Programs*). This has two advantages:

1. The problems are modified quite easily by adding new variables or contraints.
2. Solving (integer) linear problems is a field where a lot of research has been done, and there are very efficient techniques known to solve ILPs.

Motivation (4)

In the following lectures we will discuss:

- How to model (structural) multiple alignment problems as graph problems, which in turn are easy to translate into ILPs.
- Approaches to solve the ILPs (e.g., the branch-and-cut approach).
- The most important points one has to address in solving a problem with branch-and-cut:
  - identifying classes of facet-defining inequalities
  - how to solve the associated separation problems

Linear programming

We first give a short overview of linear programming and how to solve (integer) linear programs using software packages.

A *linear program (LP)* consists of a set of linear inequalities,

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots a_{1n}x_n &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots a_{2n}x_n &\leq b_2 \\ &\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots a_{mn}x_n &\leq b_m, \end{aligned}$$

together with an *objective function*

$$c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

to be optimized, i.e., minimized or maximized.

Linear programming (2)

Linear programs can be efficiently solved using the *simplex method*, developed by George Dantzig in 1947, or using *interior point methods* introduced by Khachiyan in 1979. The simplex algorithm has an exponential worst case complexity but runs quite quickly in practice.

Interior point methods were first only a proof that linear programming can be solved in polynomial time. The original algorithms were not practical. Meanwhile most LP solvers have both methods as practical routines.

Linear programming (3)

There exist powerful computer programs for solving LPs, even when huge numbers of variables and inequalities are involved.

CPLEX is the most well known and powerful commercial LP solver. `SoPlex` is part of the *ZIB Optimization Suite* (`http://zibopt.zib.de`), which is free for academic purposes.

For the rest of the lecture we will consider a LP solver as a black box which can solve linear programs very efficiently and numerically relatively stable.

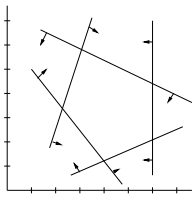
Linear programming (4)

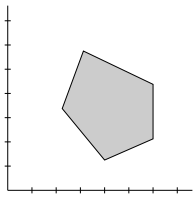
We give a small example of a LP. The inequalities of an LP describe a convex *polyhedron*, which is called a *polytope*, if it is bounded.

For example, the inequalities

$$\begin{array}{rcl} -1x_1 - 1x_2 & \leq & 5 \\ -2x_1 + 1x_2 & \leq & -1 \\ 1x_1 + 3x_2 & \leq & 18 \end{array} \qquad \begin{array}{rcl} 1x_1 + 0x_2 & \leq & 6 \\ 1x_1 - 2x_2 & \leq & 2 \end{array}$$

describe the hyperplanes and polytope depicted in this cartoon:





For example, the objective function $2x_1 - 3x_2$ takes on a maximum of 6, for $x_1 = 6$ and $x_2 = 2$, and a minimum of -9, for $x_1 = 3$ and $x_2 = 5$.

Linear programming (5)

If we are after integral solutions, we were lucky with our example. The solution of the linear program is integer. However, if we change some contraints, we are not so lucky. Assume we enter the following LP into our solver (shown in CPLEX LP format):

```
maximize  2x1-3x2
subject to
 -1x1 -   1x2 <= 5
 -2x1 + 1.5x2 <= -1
  1x1 +   3x2 <= 18
0.8x1 +   0x2 <= 6
  1x1 -   2x2 <= 2
end
```

Then the optimal solution is fractional, namely $x_1 = 7.5$ and $x_2 = 2.75$. However, we often want to constrain the variables to be *integer*, that means we want to solve an ILP.

Integer linear program

An *integer linear program (ILP)* is a linear program with the additional constraint that the variables x_i are only allowed to take on integer values.

Solving ILPs has been shown to be NP-hard. (See the book by Garey and Johnson 1979, for this and many other NP-completeness results.)

There exist a number of different strategies for approximating or solving such an ILP. These strategies usually first attempt to solve *relaxations* of the original problem, which are obtained by dropping some of the inequalities.

A very common relaxation is the *LP-relaxation* of the ILP, which is the LP obtained by dropping the integer condition.

Integer linear program (2)

In the above example we can ask CPLEX to solve an ILP by specifying

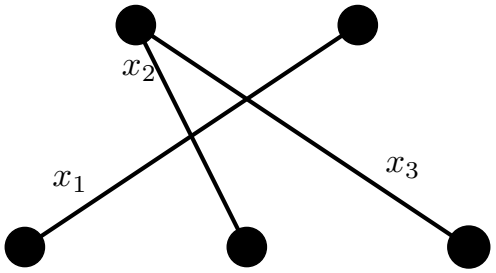
```
general
x1
x2
```

Now the optimal solution is $x_1 = 6$ and $x_2 = 2$.

Among the integer linear programs we have a special class, namely the *combinatorial optimization* problems. In those we restrict the variables to be binary. If we take an "object" into the solution then the associated variable is 1 otherwise it is 0.

Let's illustrate this on an example. Assume we are given a drawing of a bipartite graph and we want to find the largest subgraph such that no edge crosses or touches another edge in the drawing.

Integer linear program (3)



$$\max x_1 + x_2 + x_3$$
$$x_1 + x_2 \leq 1$$
$$x_1 + x_3 \leq 1$$
$$x_2 + x_3 \leq 1$$

Obviously we can only choose one of the three edges. However, solving the LP on the right gives us a solution of $x_1 = x_2 = x_3 = \frac{1}{2}$, which satisfies the contraints and maximizes the objective function. If we require the solution to be integer, only one of the three variables is set to 1.

Integer linear program (4)

In CPLEX we can do this by specifying

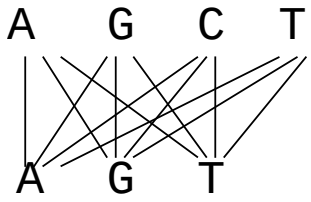
```
binary
x1
x2
x3
```

We now discuss how to model multiple sequence alignment problems as combinatorial optimization problems by first formulating them as a graph problem and then do the obvious 1-to-1 mapping between edges and variables.

The alignment graph

Given two sequences a^1 = A G C T and a^2 = A G T .

The *complete alignment graph* is the following bipartite graph $G = (V, E)$, with node set V and edge set E:

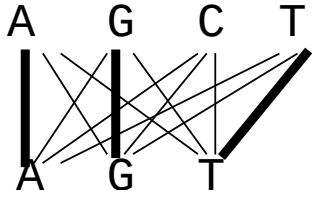


Each edge $e = (u, v)$ has a weight $\omega(e) = s(u, v)$, the score for placing v under u.

An *alignment graph* is any subgraph of the complete alignment graph.

The trace of an alignment

Given an alignment such as
A G C T
A G - T
, we say that an edge in the alignment graph is *realized*, if the corresponding positions are aligned:



The set of realized edges is called the *trace* of the alignment. An arbitrary subset $T \subseteq E$ of edges is called a *trace*, if there exists some alignment that it realizes.

Similarly, we define the (complete) alignment graph and trace for multiple alignments. For r sequences, the resulting graph will be r-partite.

Maximum-weight trace problem

Problem. Given sequences A and a corresponding alignment graph $G = (V, E)$ with edge weights ω. The maximum-weight trace problem is to find a trace $T \subseteq E$ of maximum weight.

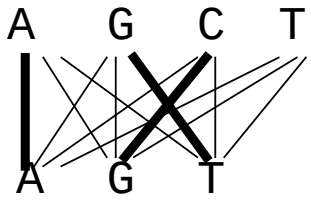
Note, that for two sequences, this can be solved in polynomial time by dynamic programming.

Characterization of traces

We have seen that an alignment can be described by a trace in the complete alignment graph $G = (V, E)$.

Question: Is *every* subset $T \subseteq E$ the trace of some alignment?

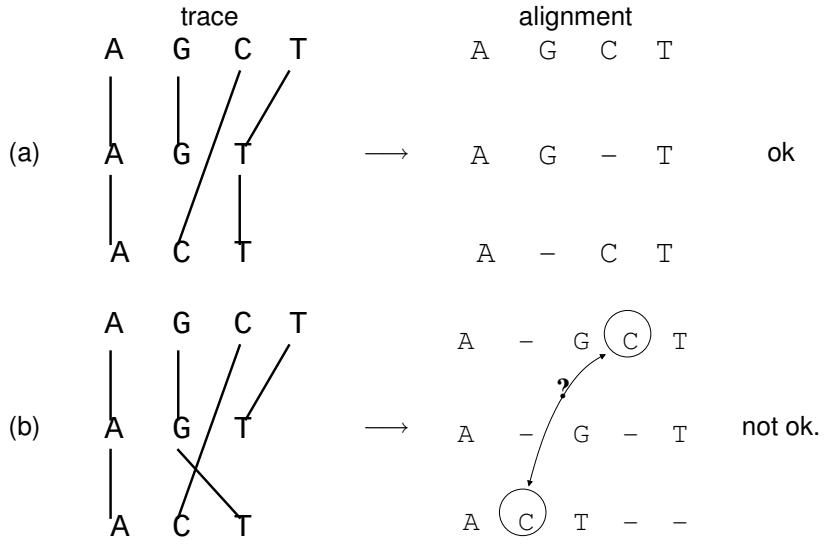
Clearly, the answer is *no*:



Goal: Characterize all legal traces.

Characterization of traces (2)

Here are two examples:



Partial orders

A binary relation $\leq$ is a *(non-strict) partial order*, if it is

1. *reflexive*, i.e., $a \leq a$,
2. *antisymmetric*, i.e., $a \leq b$ and $b \leq a$ implies $a = b$, and
3. *transitive*, i.e., $a \leq b$ and $b \leq c$ implies $a \leq c$.

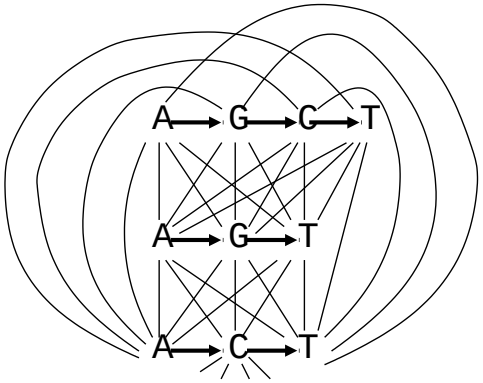
A binary relation $<$ is a *strict partial order*, if it is

1. *irrefexive*, i.e., $a \not< a$, and
2. *transitive*, i.e., $a < b$ and $b < c$ implies $a < c$.

Given a binary relation $\prec$, the *transitive closure* of $\prec$ is a binary relation $\prec^*$ such that $x \prec^* x'$ if there exists a sequence of elements $x = x_1, x_2, \ldots, x_k = x'$ with $x_1 \prec x_2 \prec \ldots x_k$.

The extended alignment graph

We define a binary relation $\prec$ on the characters of the sequences $A = \{a_j^i\}$ by writing $a_j^i \prec a_{j'}^i$, if $j' = j + 1$, and indicate the pairs (a_j^i, a_{j+1}^i) by a set H of directed edges in the alignment graph. This results in the *extended* alignment graph $G = (V, E, H)$.



Let $\prec^*$ denote the *transitive closure* of $\prec$, i.e., we write $a_j^i \prec^* a_{j'}^i$, if $j' > j$. Observe that $\prec^*$ is a strict partial order.

The extended alignment graph (2)

Consider two sets of nodes $X \subseteq V$ and $Y \subseteq V$. We define

$$X \lhd Y,$$

if and only if

$$\exists\, x \in X\ \exists\, y \in Y : x \prec y.$$

We define $\lhd^*$ to be the transitive closure of $\lhd$, that is, we write $X \lhd^* Y$, if for one of the sequences $a^p \in A$ we have that X contains a node representing a position a^p_j in a^p and Y contains a node representing another position a^p_k in a^p, with $j < k$.

In other words, we write

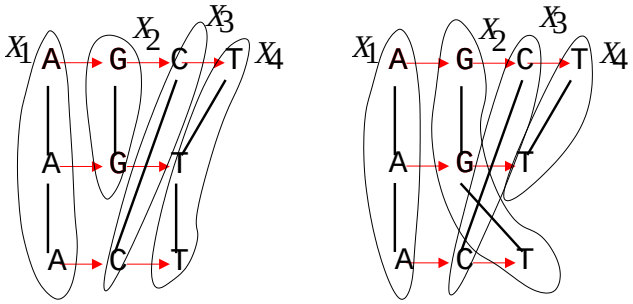
$$X \lhd^* Y,$$

if and only if

$$\exists\, x \in X\ \exists\, y \in Y : x \prec^* y.$$

The extended alignment graph (3)

Consider the two examples again and define sets $X_1, X_2, \ldots$ via the two given traces:



In (a), $X_1 \lhd^* X_2 \lhd^* X_3 \lhd^* X_4$, and $\lhd^*$ is a strict partial order.

In (b), we have $X_1 \lhd^* X_2, X_3, X_4$; $X_2 \lhd^* X_3, X_4$; and $X_3 \lhd^* X_2, X_4$. Since, e.g., $X_2 \lhd^* X_2$ does *not* hold, there is no transitivity, and thus the binary relation no partial order.

Characterization of traces

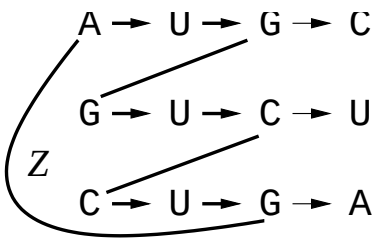
Theorem. (John Kececioglu)
Given a set of sequences A. Let $G = (V, E, H)$ be an extended alignment graph for A. A subset $T \subseteq E$ of edges is a trace, if and only if $\lhd^*$ is a strict partial order on the connected components of $G' = (V, T)$.

(Recall that a *connected component* of a graph is a maximal set of nodes $U \subseteq V$ such that any two nodes $v, u \in U$ are connected by a path of edges in the graph.)

Proof. Blackboard.

Mixed cycles

For a given choice of alignment edges we can efficiently check whether the connected components allow such a partial order by searching for a *mixed cycle* Z, which is a cycle in the extended alignment graph $G = (V, E, H)$:



A mixed cycle contains at least one arc $a \in H$ and hence at least two alignment edges $e, f \in E$.

Mixed cycles (2)

A mixed cycle Z is called *critical*, if all nodes in $Z \cap a^p$ occur consecutively in Z, for all sequences $a^p \in A$. That is, the cycle enters and leaves each sequence at most once.

We have the following result:

Lemma. A subset $T \subseteq E$ is a trace, if and only if $G' = (V, T, H)$ does not contain a critical mixed cycle.

Proof. Exercise.

Given edge weights for the alignment edges, we can reformulate the Maximum Weight Trace problem as follows:

Problem. Given an extended alignment graph $G = (V, E, H)$, find a subset $T \subseteq E$ with maximal weight such that $G = (V, T, H)$ does not contain a mixed cycle.

Integer LP for the MWT problem

How to encode the Maximum Weight Trace Problem problem as an integer LP?

Assume we are given an extended alignment graph $G = (V, E, H)$, with $E = \{e_1, e_2, \ldots, e_n\}$.

Each edge $e_i \in E$ is represented by a variable x_i, that will take on value 1, if e_i belongs to the best scoring trace, and 0, if not.

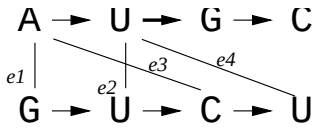
Hence, our variables are $x_1, x_2, \ldots, x_n$.

To ensure that the variables are *binary*, we add constraints $x_i \leq 1$ and $x_i \geq 0$ and require the x_i to be integer.

Additional inequalities must be added to prevent mixed cycles.

ILP for the MWT problem (2)

For example, consider:



There are three possible critical mixed cycles in the graph, one using e_1 and e_3, one using e_2 and e_3, and one using e_2 and e_4. We add the constraints

$$x_1 + x_3 \leq 1,$$

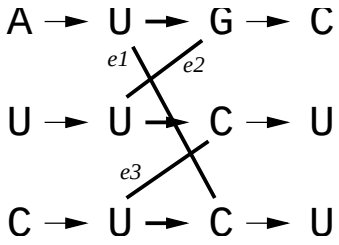
$$x_2 + x_3 \leq 1,$$

$$x_2 + x_4 \leq 1.$$

to ensure that none of the critical mixed cycles is realized.

ILP for the MWT problem (3)

For example, consider:



with three edges e_1, e_2, and e_3 that all participate in a critical mixed cycle. The constraint

$$x_1 + x_2 + x_3 \leq 2$$

prevents them from being realized simutaneously.

ILP for the MWT problem (4)

In summary, given an extended alignment graph $G = (V, E, H)$ with $E = \{e_1, e_2, \ldots, e_n\}$, and a score ω_i defined for every edge edge $e_i \in E$.

We can obtain a solution to the MWT problem by solving the following ILP:

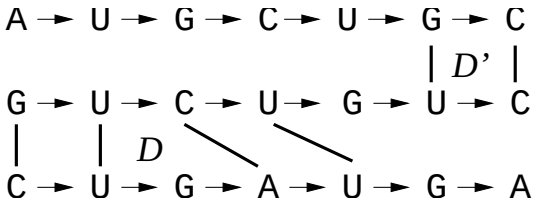
$$\begin{aligned} \max \quad & \sum_{e_i \in E} \omega_i x_i \\ \text{subject to} \quad & \sum_{e_i \in C \cap E} x_i \leq |C \cap E| - 1 && \text{for all critical mixed cycles } C \\ & x_i \in \{0,1\} && \text{for all } i = 1, \ldots, n \end{aligned}$$

Now we discuss an extension of this ILP formulation. [Remember: one advantage of ILPs is that problem variants/modifications can often be expressed quite easily.]

Block partition

Given a set of sequences $A = \{a^1, a^2, \ldots, a^r\}$. The complete alignment graph is usually too big to be useful.

Often, we are given a set of *block matches* between pairs of the sequences a^p and a^q, where a match relates a substring of a^p and a substring of a^q via a run of *non-crossing* edges (called a *block*), as shown here for two blocks D and D':



In the following, we will assume that the edges of the alignment graph $G = (V, E)$ were obtained from a set of matches, and we are given a partition of E into blocks. [Note that overlapping matches lead to a *multigraph*. Fortunately, this does not cause problems in our formulation.]

Generalized maximum trace problem (2)

Given a partition $\mathcal{D}$ of the edges of $G = (V, E)$ obtained from a set of matches. Then we require that for any given block $D \in \mathcal{D}$, either all edges in D are realized, or none. Each block D is assigned a positive weight $\omega(D)$.

Problem. Given an extended alignment graph $G = (V, E, H)$ and a partition $\mathcal{D}$ of E into blocks with weights $\omega(D)$ for all $D \in \mathcal{D}$. The *generalized maximum trace problem (GMT)* is to determine a set $M \subseteq \mathcal{D}$ of maximum total weight such that the edges in $\bigcup_{D \in M} D$ do not induce a mixed cycle on G.

Instead of having a variable for every edge in an extended alignment graph, we now have a variable for every set of the partition $\mathcal{D}$.

Otherwise the ILP remains the same.

ILP for the GMT

We define a surjective function $v : E \to \mathcal{D}$, which maps each edge $e \in E$ to the block $d \in \mathcal{D}$ in which e is contained and define

$$v(X) = \bigcup_{e \in X} v(e) \qquad \text{for } X \subseteq E \ .$$

It is now easy to formulate GMT as an integer linear program. For every $d \in \mathcal{D}$ we have a binary variable $x_d \in \{0,1\}$ indicating whether d is in the solution or not. Then the GMT-problem can be written as:

$$\begin{aligned}
\max \quad & \sum_{d \in \mathcal{D}} \omega_d \cdot x_d & & \\
\text{s.t.} \quad & \sum_{d \in v(C \cap E)} x_d \leq |v(C \cap E)| - 1 & & \forall \text{ critical mixed cycles } C \text{ in } G \\
& x_d \in \{0,1\} & & \forall d \in \mathcal{D}
\end{aligned}$$

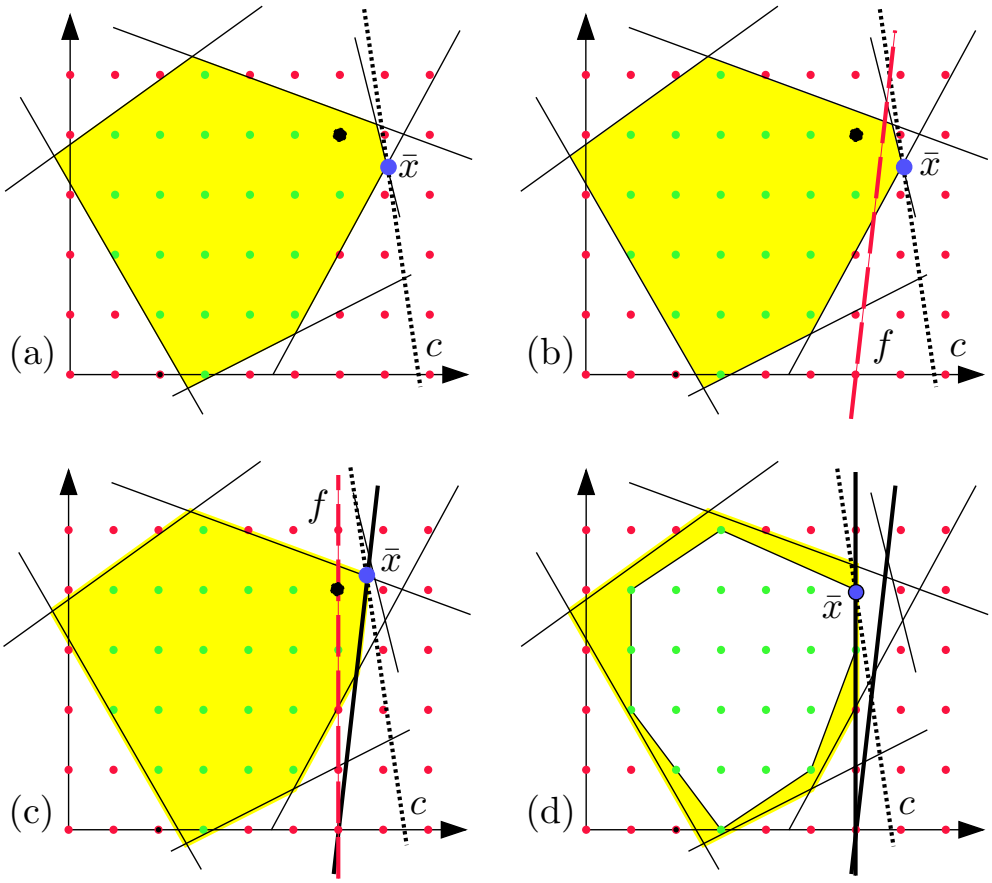
Solving the ILP using branch-and-cut

Solving ILPs is a main topic in combinatorial optimization. We will take a brief look at the *branch-and-cut* approach.

Branch-and-cut makes use of two techniques:

- **Cutting planes:** to solve an ILP, one considers relaxations of the problem (usually the LP-relaxation) and repeatedly "cuts" away parts of the polytope (by adding new constraints) in the hope of obtaining an integer solution. To find cutting planes one has to solve the *separation problem*, that means find a violated, *valid* inequality of the ILP.
- **Branch-and-bound:** an enumeration tree of all possible variable settings is partially traversed, computing local upper bounds and global lower bounds, which are used to avoid parts of the tree that cannot produce the optimal value.

Solving the ILP using branch-and-cut (2)



Adding cutting planes. The last cutting plane is facet-defining and leads to the optimal integer solution.

Solving the ILP using branch-and-cut (3)

The convex hull of all feasible incidence vectors (the green points in the previous figure) forms the *problem polytope P* (the inner, white polytope in the previous figure). If one considers the inequalities of the ILP, they generally describe a larger polytope (the yellow polytope), although this polytope does not contain an infeasible integer point (the red points).

If the solution $\bar{x}$ of the LP-relaxation is integral, it corresponds to a feasible incidence vector that represents an optimal solution. Otherwise we search for a *valid* inequality $fx \leq f_0$ that "cuts off" the solution $\bar{x}$, i. e.,

$$\begin{aligned} fy &\leq f_0 \qquad \text{for all } y \in P \text{ and} \\ f\bar{x} &> f_0 \ . \end{aligned}$$

The set $\{x \mid fx = f_0\}$ is called a *cutting plane*.

Solving the ILP using branch-and-cut (4)

The search for a cutting plane is called the *separation problem*. Any cutting plane found is added to the linear program and the linear program is solved again.

There is a special class of cutting planes we are interested in, namely the *facets* of the problem polytope. As we know, facets of a d-dimensional polytope are faces of dimension $d-1$. They are in a sense the "best" cutting planes since they directly bound the problem polyhedron. Hence, it is important to *identify the classes of facet-defining inequalities*.

Solving the ILP using branch-and-cut (5)

We generally compute cutting planes for two reasons:

1. A class of contraints in our ILP formulation is too large (i.e., exponentially large) to write down (we relax the LP-relaxation further).
2. The inequalities in the original ILP formulation are not sufficient to yield an integer solution. Hence we search for valid inequalities that cut off those fractional solutions.

For example consider the number of mixed cycles in the MWT problem. It grows exponentially with the size of the graph. Instead of writing them all down, we relax the problem first by omitting them, and then compute cutting planes by checking whether they are violated. [Indeed we do something slightly different, since the mixed cycle inequalities do *not* always describe facets of the MWT-polytope.]

ILP using branch-and-cut (6)

How often do we have to repeat that? Are we guaranteed to find a solution? The following theorem by Grötschel, Lovász, and Schrijver gives a partial answer:

Theorem. For any proper class of polyhedra, the optimization problem is polynomially solvable if and only if the separation problem is polynomially solvable.

This basically says, that for NP-complete problems there is little hope to guarantee a quick termination of the cutting plane algorithm.

ILP using branch-and-cut (7)

Either we just do not know enough classes of facet-defining inequalities, or we cannot solve the separation problem for one of these classes in polynomial time.

Hence we repeat the cutting step either until an integer solution is found that fulfills all constraints (which would be optimal for the original problem), or until we get stuck, that means we cannot find a violated inequality for *any* of our classes of valid inequalities.

ILP using branch-and-cut (8)

In this case we **branch**. That is, we choose a variable x_i and solve two sub-cases, namely the case $x_i = 0$ and the case $x_i = 1$. Repeated application produces a enumeration tree of possible cases.

We call an upper bound for the original ILP *local*, if it is obtained from considering a subproblem in the enumeration tree.

If the solution found for a subproblem is feasible for the original problem and has a higher score than any solution found so far, then it is recorded and its value becomes the new *global lower bound* for the original objective function. Remember that the feasibility of a new solution has to be carefully checked, since we fixed a number of variables on our way to the subproblem.

ILP using branch-and-cut (9)

Subsequently, we only pursue subproblems whose local upper bound is greater or equal to the global lower bound.

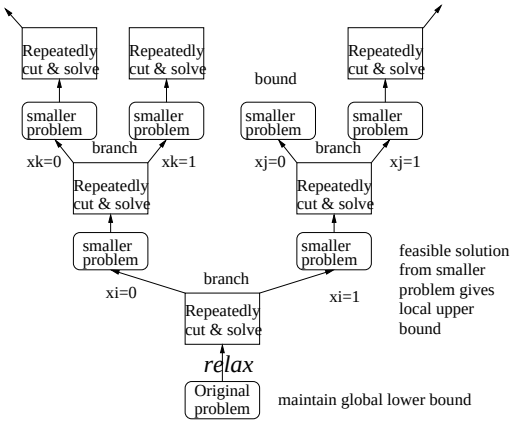
This is an example of the *branch-and-bound* paradigm for solving hard combinatorial problems.

Branch-and-bound is a divide-and-conquer approach to solving a problem by dividing it into smaller problems. The *local* solution of a subproblem gives rise to a lower bound for the best possible score for the original problem, if the solution of the subproblem also solves the original problem.

The highest such local score attained so far gives rise to a *global* lower bound for the original problem and is used to skip parts of the enumeration tree in which the current global lower bound can't be beaten.

ILP using branch-and-cut (10)

General strategy:



The details are a bit more involved and are skipped here. We now take the initial MWT problem as an example to point out the important steps in branch-and-cut algorithms.

Solving the MWT

Recall the ILP for the MWT. We can obtain a solution to the MWT problem by solving the following ILP:

$$\begin{aligned} \max \quad & \sum_{e_i \in E} \omega_i x_i & & \\ \text{subject to} \quad & \sum_{e_i \in C \cap E} x_i \leq |C \cap E| - 1 & & \text{for all critical mixed cycles } C \\ & x_i \in \{0,1\} & & \text{for all } i = 1, \ldots, n \end{aligned}$$

We showed before that this ILP describes the solution to the Maximum Weight Trace problem. The first step is to have a closer look at the MWT-polytope.

Solving the MWT (2)

Let $\mathcal{T} := \{T \subseteq E \mid T \text{ is a trace}\}$ be the set of all feasible solutions. We define the *MWT polytope* as the convex hull of all incidence vectors of E that are feasible, i. e.,

$$P_{\mathcal{T}}(G) := \text{conv}\{\chi^T \in \{0,1\}^{|E|} \mid T \in \mathcal{T}\} \ ,$$

where the *incidence vector* χ^T for a subset $T \subseteq E$ is defined by setting $\chi^T_e = 1$ if $e \in E$ and setting $\chi^T_e = 0$ if $e \notin E$.

We have a closer look at the facial structure of the polytope, that means we try to *identify facet-defining classes of inequalities*. The following theorem is our main tool.

Identifying facet-defining classes of polytope

Theorem. ("facet theorem") (Nemhauser, Trotter, 1973)
Let $P \subseteq \mathbb{Q}^d$ be a full-dimensional polyhedron. If F is a (nonempty) face of P then the following assertions are equivalent.

1. F is a facet of P.
2. $\dim(F) = \dim(P) - 1$, where $\dim(P)$ is the maximum number of affinely independent points in P minus one.
3. There exists a valid inequality $c^T x \leq c_0$ with respect to P with the following three properties:

(a) $F = \{x \in P \mid c^T x = c_0\}$

(b) There exists a vector $\hat{x} \in P$ such that $c^T \hat{x} < c_0$.

(c) If $a^T x \leq a_0$ is a valid inequality for P such that $F \subseteq \bar{F} = \{x \in P \mid a^T x = a_0\}$ then there exists a number $\lambda \in \mathbb{Q}$ such that $a^T = \lambda \cdot c^T$ and $a_0 = \lambda \cdot c_0$.

Identifying facet-defining classes of polytope (2)

Assertions 2 and 3 provide the two basic methods to prove that a given inequality $c^T x \leq c_0$ is facet-defining for a polyhedron P.

The first method (Assertion 2), called the *direct* method, consists of exhibiting a set of $d = \dim(P)$ vectors $x_1, \ldots, x_d$ satisfying $c^T x_i = c_0$ and showing that these vectors are affinely independent.

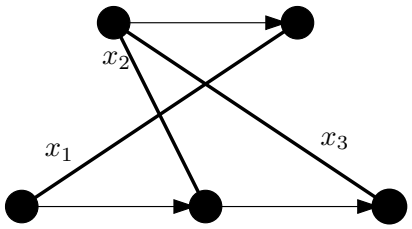
The *indirect* method (Assertion 3) is the following: We assume that

$$\{x \mid c^T x = c_0\} \subseteq \{x \mid a^T x = a_0\}$$

for some valid inequality $a^T x \leq a_0$ and prove that there exists a $\lambda > 0$ such that $a^T = \lambda \cdot c^T$ and $a_0 = \lambda \cdot c_0$.

Clique inequalities

Now we describe a class of valid, facet-defining inequalities for the MWT problem, focusing first on the *pairwise case*. In the case of two sequences, consider the following extended alignment graph:



This gives rise to the following set of inequalities:

$$x_1 + x_2 \leq 1,\ x_1 + x_3 \leq 1,\ x_2 + x_3 \leq 1$$

However, it is clear that only one of the three edges can be realized by an alignment. Hence, inequality $x_1 + x_2 + x_3 \leq 1$ is *valid* and more stringent. Indeed it cuts off the fractional solution $x_1 = x_2 = x_3 = \frac{1}{2}$.

Clique inequalities (2)

If $C \subseteq E$ is a set of alignment edges such that each pair forms a mixed cycle, it is called a *clique* (since it forms a clique in the *conflict graph*).

The conflict graph of a combinatorial optimization problem has a node for each object and an edge between pairs of conflicting objects). In general the *clique* inequalities

$$\sum_{e \in C} x_e \leq 1$$

are valid for the MWT problem.

Are they also facet-defining for the MWT polytope?

Theorem.
Let $C \subseteq E$ be a *maximal* clique. Then the inequality $\sum_{e \in C} x_e \leq 1$ is facet-defining for $P_{\mathcal{T}}(G)$.

Clique inequalities (3)

Proof.
We choose the direct way, which means we have to find n affinely independent vectors satisfying $\sum_{e \in C} x_e = 1$. This can be easily achieved. Assume without loss of generality that $|E \setminus C| \neq \varnothing$. We first construct $|C|$ many solutions by choosing a single edge in C.

Then for each edge $e \notin C$ there must be an edge $f \in C$ which does not form a mixed cycle with e (otherwise C is not maximal). Hence we can construct a set of solutions $\{e, f\}$, $\forall e \notin C$. This means we have for all n edges a solution satisfying the clique inequality with equality, and they are clearly affinely independent. ■

Clique inequalities (4)

But how do we efficiently find violated clique inequalities? How do we solve the separation problem? We define the following relation on edges:

Definition.
Let $K_{p,q}$ be the complete bipartite graph with nodes $x_1, \ldots, x_p$ and $y_1, \ldots, y_q$. Define the strict partial order '$\prec$' on the edges of $K_{p,q}$ as follows:

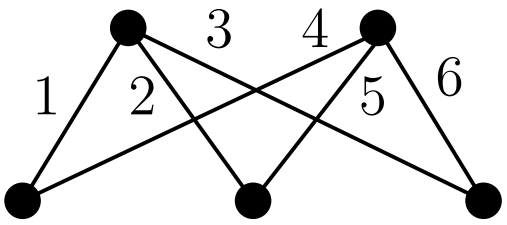
$$e = (x_i, y_j) \prec f = (x_k, y_l) \text{ iff}$$

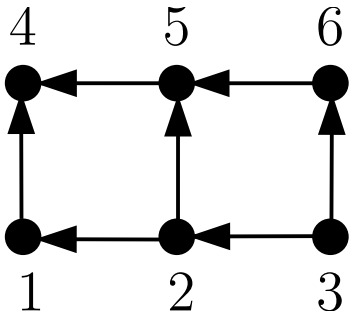
$$(i > k \text{ and } j \leq l) \text{ or } (i = k \text{ and } j < l).$$

Observe that for two sequences the alignment graph (V, E) is a subgraph of $K_{p,q}$ and that two edges e and f form a mixed cycle in the input graph iff either $e \prec f$ or $f \prec e$.

Clique inequalities (5)

Definition.
Let $PG(K_{p,q})$ be the $p \times q$ directed grid graph with arcs going from right to left and from bottom to top. Row r, $1 \leq r \leq p$ of $PG(K_{p,q})$ contains q nodes which correspond from left to right to the q edges that go between node x_{p-r+1} and node $y_1, \ldots, y_q$ in $K_{p,q}$. We call $PG(K_{p,q})$ the *pairgraph* of $K_{p,q}$ and we call a node of the pairgraph *essential* if it corresponds to an edge in E.





Clique inequalities (6)

The graph $PG(K_{p,q})$ has exactly one source and one sink and there is a path from node n_2 to node n_1 in $PG(K_{p,q})$ iff $e_1 \prec e_2$ for the corresponding edges e_1, e_2 in $K_{p,q}$.

Lemma.
Let $P = n_1, \ldots, n_{p+q}$ be a source-to-sink path in $PG(K_{p,q})$ and let $e_1, \ldots, e_l$, $l \leq p + q$, be the edges in E that correspond to essential nodes in P. Then $e_1, \ldots, e_l$ is a clique of the input extended alignment graph if $l \geq 2$. Moreover, every maximal clique in the input extended alignment graph can be obtained in this way.

Clique inequalities (7)

Proof.
For any two nodes n_i and n_j in $PG(K_{p,q})$ with $i > j$ the corresponding edges e_i and e_j are in relation $e_i \prec e_j$ and hence form a mixed cycle in G. Thus $\{e_1, \ldots, e_l\}$ is a clique of G. Conversely, the set of edges in any clique C of G is linearly ordered by $\prec$ and hence all maximal cliques are induced by source-to-sink paths in $PG(K_{p,q})$. ■

Clique inequalities (8)

We can now very easily use the pairgraphs for each pair of sequences to separate the clique inequalities.

Assume the solution $\bar{x}$ of the current LP-relaxation is fractional. Our problem is to find a clique C which violates the clique inequality

$$\sum_{e \in C \cap E} \bar{x}_e \leq 1 \ .$$

Assign the cost $\bar{x}_e$ to each essential node v_e in $PG(K_{p,q})$ (essential nodes are the nodes that correspond to the edges in E) and 0 to non-essential nodes.

Clique inequalities (9)

Then compute the longest source-to-sink path C in $PG(K_{p,q})$. If the cost of C is greater than 1, i.e.,

$$\sum_{e \in C \cap E} \bar{x}_e > 1$$

we have found a violated clique inequality.

Since $PG(K_{p,q})$ is acyclic, such a path can be found in polynomial time.

[Caution: We will not go deeper into this, but it is necessary to make a sparse version of the PG in the case of a non-complete bipartite graph. This has to be done such that its size ist still polynomial and each path encodes a maximal clique. Nevertheless, the trick with essential and non-essential nodes will work and leads to correct separation results.]

Mixed cycle inequalities (10)

Now we describe how to solve the separation problem for the mixed-cycle inequalities. Assume the solution $\bar{x}$ of the linear program is fractional.

First assign the cost $1 - \bar{x}_e$ to each edge $e \in E$ and 0 to all $a \in H$. Then we compute for each node $s_{i,j}$, $1 \leq i \leq k$, $1 \leq j < n_i$ the shortest path from $s_{i,j+1}$ to $s_{i,j}$. If there is such a shortest path P, and its cost is less than 1, i.e.,

$$\sum_{e \in P} (1 - \bar{x}_e) < 1 \ ,$$

we have found a violated inequality, namely

$$\sum_{e \in P} \bar{x}_e > |P \cap E| - 1 \ ,$$

since P together with the arc $(s_{i,j}, s_{i,j+1})$ forms a mixed cycle.

RNA structure alignment

Finally, we give an ILP formulation for combined sequence structure alignment. This shows again how we can easily extend the basic MWT formulation to a more complicated problem. We will give some classes of facet-defining inequalities.

In addition we demonstrate how to conduct an indirect proof that one specific class is facet-defining.

Furthermore, we will get to know the concept of *Lagrangian relaxation* to find solutions for ILP problems.

But first we introduce the term *structured sequence*.

Structured sequences

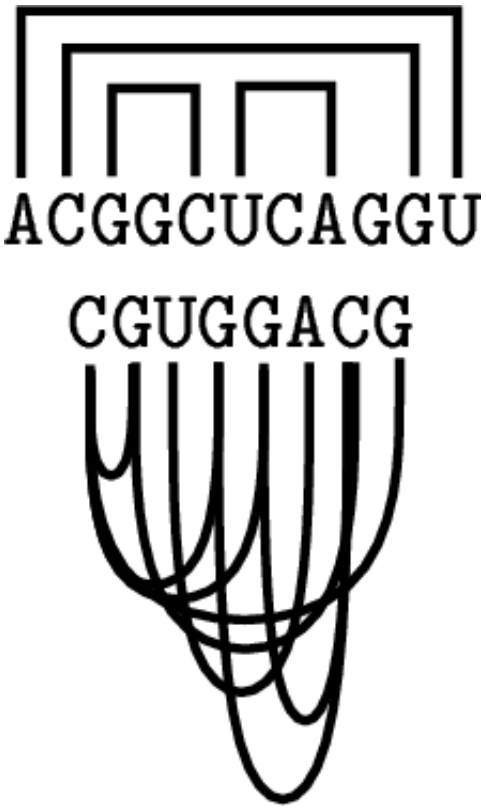
Given a set of sequences $A = \{a^1, \ldots, a^r\}$. How can we use information about their secondary structures to obtain biologically better alignments?

Let a^t be a sequence of length n. An *interaction* is a pair (i, j) of positions in a^t for $1 \leq i < j \leq n$.[1] Two interactions (i, j) and (k, l) are in *conflict*, if $\{i, j\} \cap \{k, l\} \neq \emptyset$.

Structured sequences (2)

A set P of non-conflicting interactions for a^t is called a *(secondary) structure* for a^t, and we call (a^t, P) a *structured sequence*. If P contains conflicts we call (a^t, P) an *annotated sequence*.

The figure shows a structured and an annotated sequence.



Structural alignment

Given a set of annotated sequences

$$A = \{(a^1, P_1), (a^2, P_2), \ldots, (a^r, P_r)\}.$$

A *structural multiple sequence alignment* of A is a set

$$\hat{A} = \{(\hat{a}^1, \hat{P}_1), \ldots, (\hat{a}^r, \hat{P}_r)\}$$

of structured sequences such that:

1. $\{\hat{a}^1, \hat{a}^2, \ldots, \hat{a}^r\}$ is a multiple sequence alignment for $\{a^1, a^2, \ldots, a^r\}$, and
2. for all $t \in \{1, 2, \ldots, r\}$ and all $(i, j) \in \hat{P}_t$, we have

$$(i - \#\text{gaps}(\hat{a}_1^t, \ldots, \hat{a}_i^t), j - \#\text{gaps}(\hat{a}_1^t, \ldots, \hat{a}_j^t)) \in P_t.$$

[1] We potentially require a minimum hairpin loop length of three bases for RNA, i. e., $j - i > 3$.

The second condition ensures that the structural msa only contains interactions that are present in the input.

Scoring a structural alignment

Goal: Define a scoring scheme that reflects the biological relatedness of the sequences under consideration, taking their secondary structure into account.

In general terms, a scoring function simply assigns a real number to every possible structural alignment.

In practice, scoring functions usually consist of a weighted sum of two parts, one that evaluates the alignment of the characters in the sequences and one that evaluates the alignment of the interactions in the secondary structure.

In the pairwise case, for example, one could use a pairwise sequence alignment score *sim* to score the sequence alignment and then define an *interaction score*

$$isim : \Sigma^4 \to \mathbb{R}$$

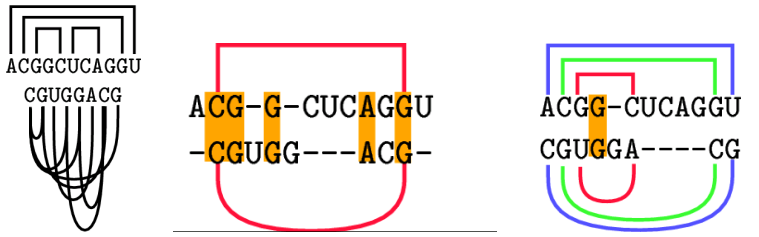
that assigns a score to two pairs of aligned characters that are defined by two matching interactions (where Σ denotes the alphabet of characters, excluding '$-$').

Scoring a structural alignment (2)

The following structural alignment score function $rna(\hat{A})$ can be used to identify pairwise structural alignments of RNA sequences that have both high sequence similarity and high structure conservation:

$$rna(\hat{A}) = \sum_{i=1}^{L} sim(\hat{a}_i^1, \hat{a}_i^2) + \sum_{\substack{(j,l) \in \hat{P}_1, (q,r) \in \hat{P}_2 \\ (q,r) = (j,l)}} isim(\hat{a}_j^1, \hat{a}_l^1, \hat{a}_q^2, \hat{a}_r^2).$$

Example:



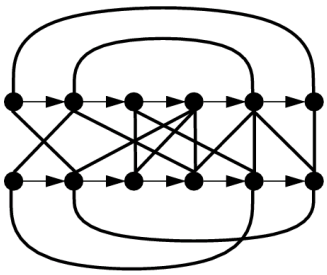
interactions | alignment with high sequence score | alignment with high interaction score

Structural alignment graph

We consider now the case of two sequences, but all definitions hold for $r > 2$. Given a set of annotated sequences $A = \{(a^1, P_1), \ldots, (a^r, P_r)\}$. Let $G = (V, E, H)$ denote a extended alignment graph associated with $\{a^1, \ldots, a^r\}$.

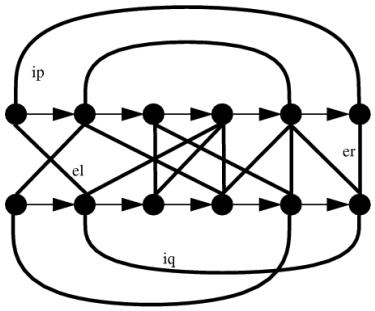
A *structural (extended) alignment graph* $G' = (V, E, H, I)$ is obtained by introducing a set of new edges I that represent all possible interactions.

Example:



Interaction matches

Let i_p and i_q denote two interaction edges in a structural alignment graph that are contained in the secondary structures of two different sequences. Given a trace T we say that i_p *matches* i_q, if the two alignment edges e_l and e_r joining the two left respectively two right nodes of i_p and i_q are *realized*, i.e., if $e_l, e_r \in T$.



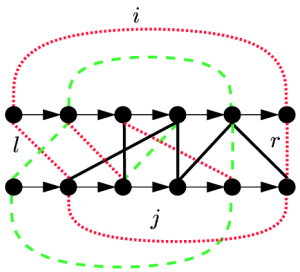
We call any such set $\{i_p, i_q, e_l, e_r\}$ an *interaction match*.

Structural traces

Let $G = (V, E, H, I)$ be a structural alignment graph for a set of structured sequences $A = \{(a^1, P_1), \ldots, (a^r, P_r)\}$. A *structural trace* of G is a pair (T, B) with $T \subseteq E$ and $B \subseteq I$ such that

- the induced subgraph $(V, T \cup H)$ does not contain a critical mixed cycle, and
- there are no two conflicting edges in B.
- each edge in B is part of a realized interaction match

Here are two possible structural traces:



Scoring a structural trace

Let (T, B) be a structural trace. As for conventional traces, each edge $e \in T$ is given a weight $\omega(e)$ which is simply the score for aligning the two corresponding symbols.

Any interaction match $\{i_p, i_q, e_l, e_r\}$ is completely specified by the two edges e_l and e_r and so we can denote it by m_{lr} and assign a weight $\omega(l, r)$ to it. Let M denote the set of all possible interaction matches, i. e.,

$$M = \{m_{lr} \mid m_{lr} = \{i_p, i_q, e_l, e_r\} \text{ is a possible interaction match in } G\} \ .$$

We define the *score of a structural trace* as:

$$S((T, B)) = \sum_{e \in T} \omega(e) + \sum_{\substack{i_p, i_q \in B,\, e_l, e_r \in T \\ \{i_p, i_q, e_l, e_r\} \in M}} \omega(l, r).$$

We can find the maximal scoring trace by solving an ILP.

Maximal scoring structural trace

Given a structural alignment graph $G = (V, E, H, I)$, the score ω_i for realizing an edge $e_i \in E$ and the score ω_{lr} for realizing an interaction match m_{lr}.

The problem of maximizing the score of a structural trace can be formulated as the following ILP:

$$
\begin{aligned}
\max \quad & \sum_{e_i \in E} \omega_i x_i + \sum_{m_{ij} \in M} \omega_{ij} y_{ij} & \\
\text{s.t.} \quad & \sum_{e_i \in C \cap E} x_i \leq |C \cap E| - 1 & \text{for all critical mixed cycles } C \\
& \sum_j y_{ij} \leq x_i & \text{for all } i \\
& \sum_i y_{ij} \leq x_j & \text{for all } j \\
& x_i, y_{ij} \in \{0, 1\} & \text{for all variables}
\end{aligned}
$$

Maximal scoring structural trace (2)

Lemma.
Any solution to this ILP corresponds to a structural trace.

Maximal scoring structural trace (3)

Proof.
The set of all variables x_i with $x_i = 1$ define a set of edges $T \subseteq E$ and the critical mixed cycle inequalities ensure that T is a trace.

The set of all variables y_{ij} with $y_{ij} = 1$ define a set of matched interactions $B \subseteq I$. To establish that B does indeed give rise to a structural trace, note that the remaining constraints in the ILP ensure that

- an interaction match m_{lr} can only be realized if both e_l and e_r are realized, and
- only one interaction match can "use" any specific alignment edge e_i as its left- or right-connecting edge, and so no conflicts can arise. ■

Facet-defining inequalities

Given a subset of interaction matches M' of M let $I(M')$ be all interaction edges of the interaction matches in M', that is

$$I(M') = I \cap \bigcup_{m_{l,r} \in M'} m_{l,r}.$$

The goal is to find the structural trace $(E', I(M'))$, $E' \subseteq E$, $M' \subseteq M$ with maximal weight.

Facet-defining inequalities (2)

The set of realized alignment edges and interaction matches can be represented by an $|E \cup M|$-dimensional incidence vector χ^A. Let

$$\mathcal{R} := \{A = (E', M'), E' \subseteq E, M' \subseteq M \mid (E', I(M')) \text{ is a structural trace of } G\}$$

be the set of all feasible solutions.

We define the *SMT polytope* of G as the convex hull of the incidence vectors of all feasible solutions, i. e.,

$$P_{\mathcal{R}}(G) := \text{conv}\{\chi^A \in \{0,1\}^{|E \cup M|} \mid A \in \mathcal{R}\}.$$

Facet-defining inequalities (3)

First we state some basic results about the SMT polytope and then define four non-trivial classes of valid inequalities and show in which case they are facet-defining.

Lemma.
Let $G = (V, E, H, I)$ be a SEAG with n alignment edges and m interaction matches. Then

- $P_{\mathcal{R}}(G)$ is full-dimensional and
- the inequality $x_i \leq 1$ is facet-defining iff there is no $e_j \in E$ in conflict with e_i.

Proof. (Exercise). Hint: find a sufficient number of affinely independent vectors to prove both parts.

Facet-defining inequalities (4)

The following lemma handles other trivial inequalities.

Lemma.
Let $G = (V, E, H, I)$ be a SEAG with n alignment edges and m interaction matches.

1. The inequality $x_i \geq 0$ is facet-defining iff e_i is not contained in an interaction match.
2. For each interaction match $m_{i,j}$ the inequality $y_{ij} \geq 0$ is facet-defining.

Proof. (Exercise)

Now we come to a more interesting class of inequalities, the *interaction inequalities*. If M_i is the set of interaction matches that contain e_i then we can write them as

$$\sum_{m_{i,j} \in M_i} y_{i,j} - x_i \leq 0 \ .$$

Facet-defining inequalities (5)

Theorem.
Let $G = (V, E, H, I)$ be a SEAG. Let e_i be an alignment edge. Then the interaction inequality

$$\sum_{m_{i,j} \in M_i} y_{i,j} - x_i \leq 0$$

is facet-defining for $P_{\mathcal{R}}(G)$.

Facet-defining inequalities (6)

Proof.
Denote the interaction inequality by $c^T \binom{x}{y} \leq c_0$. Obviously condition 3 (b) of the "facet theorem" holds because for the incidence vector χ_i of the set $\{e_i\}$ holds $c^T \chi_i < c_0$. Therefore it is sufficient to show that every valid inequality $a^T x \leq a_0$ with

$$\{x \mid c^T x = c_0\} \subseteq \{x \mid a^T x = a_0\}$$

is—up to a multiplicative factor—equal to $c^T x \leq c_0$.

Assume that $\{x \mid c^T x = c_0\} \subseteq \{x \mid a^T x = a_0\}$. Since $c_0 = 0$ it follows that $a_0 = 0$.

The incidence vectors $\chi^{\{e\}}$ of the $|E| - 1$ sets $\{e\}$, $\forall e \in E \setminus \{e_i\}$ fulfill $c^T \chi^{\{e\}} = a^T \chi^{\{e\}} = 0$. Hence, $a_e = 0$, $\forall e \in E \setminus \{e_i\}$.

Similarly the $|M| - |M_i|$ incidence vectors $\chi^{A_{m_{l,r}}}$ of the sets $A_{m_{l,r}} = m_{l,r}$, $\forall m_{l,r} \in M \setminus M_i$ fulfill $c^T \chi^{A_{m_{l,r}}} = a^T \chi^{A_{m_{l,r}}} = 0$. Hence, $a_{m_{l,r}} = 0$, $\forall m_{l,r} \in M \setminus M_i$.

Facet-defining inequalities (7)

Proof (continued):
The sets $A_{m_{i,j}} = m_{i,j}$, $\forall m_{i,j} \in M_i$ form feasible solutions whose incidence vectors $\chi^{A_{m_{i,j}}}$ satisfy $c^T \chi^{A_{m_{i,j}}} = 0$ and therefore $a^T \chi^{A_{m_{i,j}}} = 0$.

If one subtracts $a^T \chi^{A_{m_{i,j}}} = 0$ from $a^T \chi^{A_{m_{i,j'}}} = 0$, $\forall m_{i,j'} \in M_i \setminus \{m_{i,j}\}$ this yields $a_{m_{i,j}} = \cdots = a_{m_{i,j'}}$, because we have just shown that all other coefficients except a_{e_i} are zero.

Since $a_{m_{i,j}} = -a_{e_i}$ we can choose $\lambda = \frac{c_{m_{i,j}}}{a_{m_{i,j}}}$ which yields $\lambda \cdot a^T = c^T$.

Applying the "facet theorem" proves that the interaction inequalities are indeed facet-defining for $P_{\mathcal{R}}(G)$. ■

Summary

- We can formulate an alignment problem as a graph-theoretic problem, the maximum weight trace problem.
- This is a suitable basis for translating the problem into an ILP. The variables are edges and forbidden combinations of variables can be easily detected (by searching mixed cycles).
- The formulation can be easily extended to handle block alignment or sequence/structure alignments.
- In general, ILP problems are NP-hard. One possible way of solving is the branch-and-cut approach.
- In branch-and-cut it is important to describe the problem polytope as well as possible. That means should identify facet-defining inequalities and find efficient separation routines for them.
- We saw a central theorem that helps to prove whether an inequality is facet-defining or not.