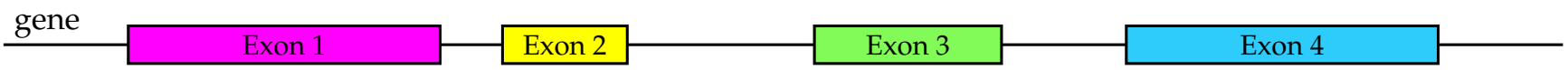
A Novel Combinatorial Method for Estimating Transcript Expression with RNA-Seq: Finding a Bounded Number of Paths

Alexandru I. Tomescu[1], Anna Kuosmanen[1], Romeo Rizzi[2], Veli Mäkinen[1]

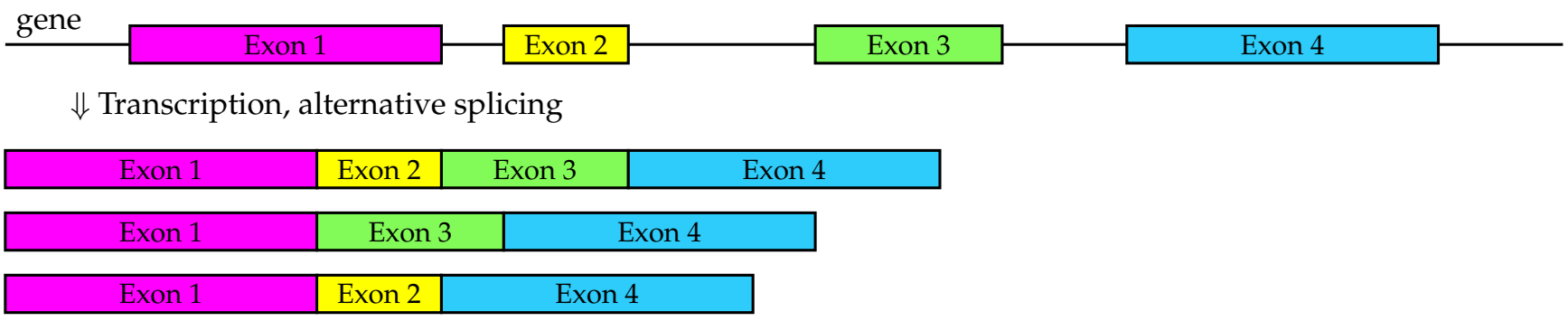
[1]*Helsinki Institute for Information Technology HIIT, Department of Computer Science, University of Helsinki, Finland*
[2]*Department of Computer Science, University of Verona, Italy*

HiTSeq
July 20, 2013

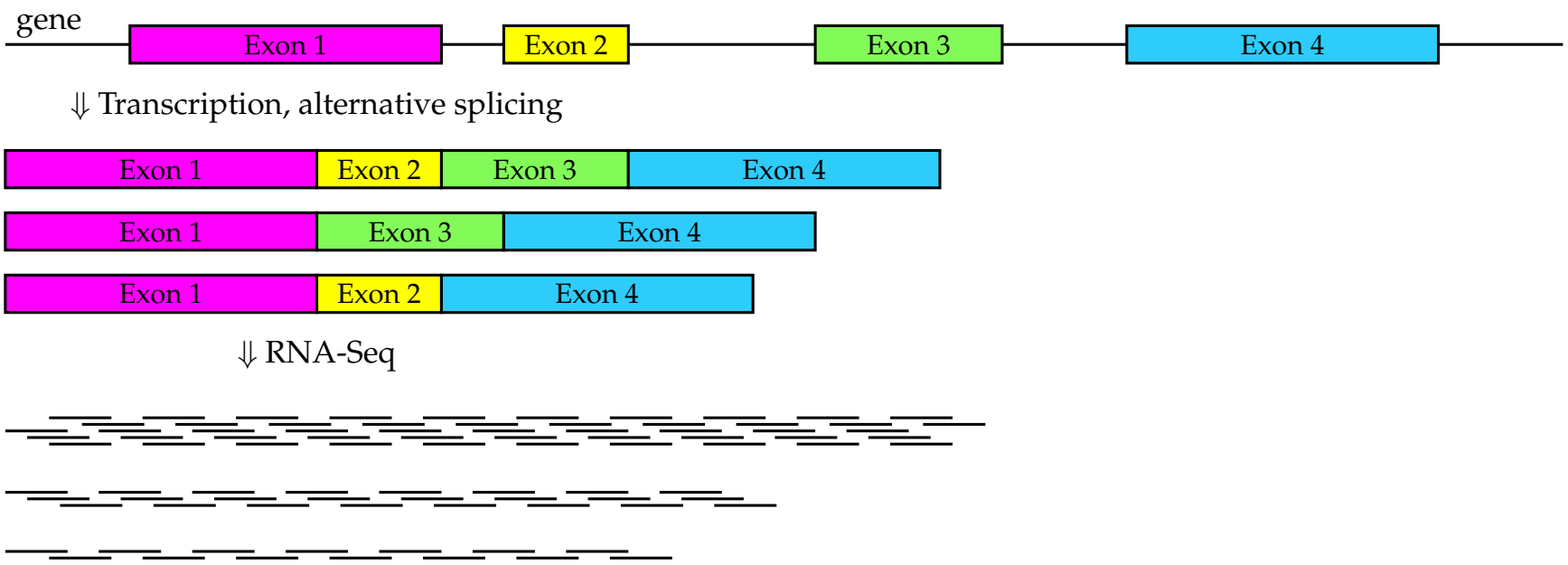
THE BIOLOGICAL PROBLEM



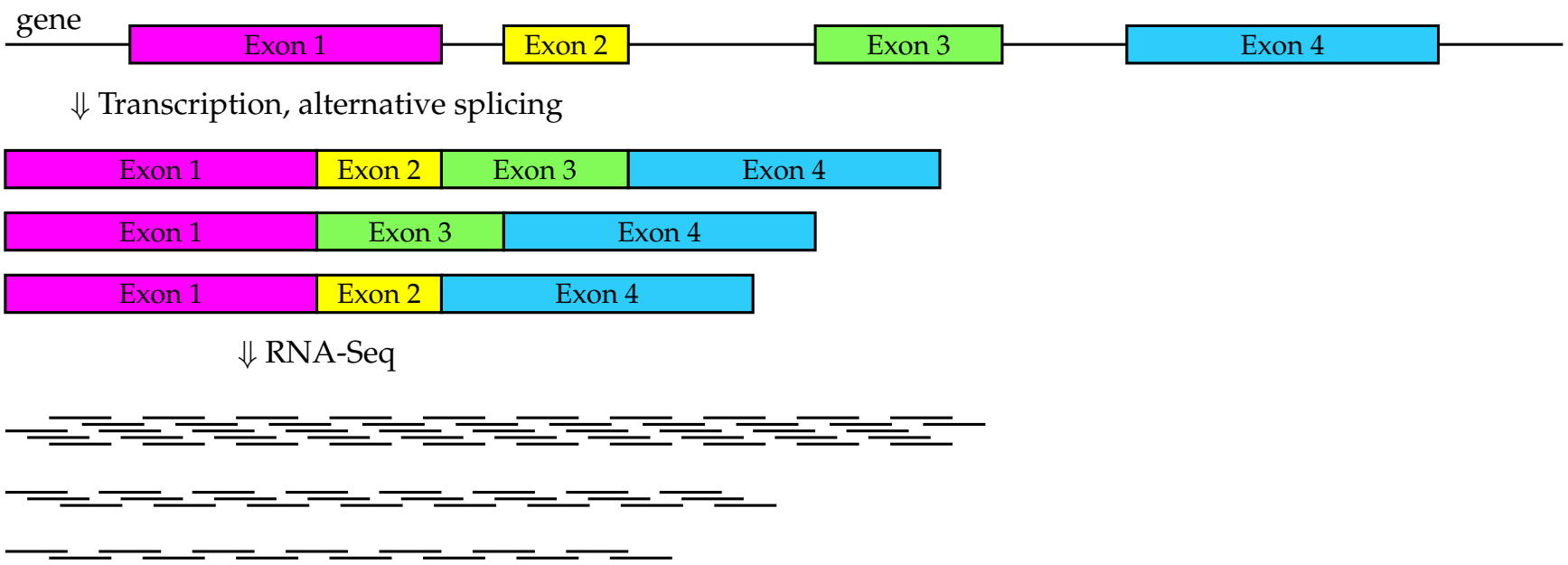
THE BIOLOGICAL PROBLEM



The biological problem



THE BIOLOGICAL PROBLEM



The problem: Assemble the transcripts and estimate their expression levels using only the RNA-Seq reads

Existing methods

Genome-independent:

- TransABySS ('10), Trinity ('11), Oases ('12)

Existing methods

Genome-independent:

- TransABySS ('10), Trinity ('11), Oases ('12)

Genome-guided:

- Annotation-free:
  Scripture ('10), TRIP ('12),

Existing methods

Genome-independent:

- TransABySS ('10), Trinity ('11), Oases ('12)

Genome-guided:

- Annotation-free:
  Scripture ('10), TRIP ('12),
- Annotation-guided:
  Cufflinks ('10), IsoLasso ('11), SLIDE ('11), iReckon ('12), CLIIQ ('12)

Existing methods

Genome-independent:

- TransABySS ('10), Trinity ('11), Oases ('12)

Genome-guided:

- Annotation-free:
  Scripture ('10), TRIP ('12),
- Annotation-guided:
  Cufflinks ('10), IsoLasso ('11), SLIDE ('11), iReckon ('12), CLIIQ ('12)

Existing methods

Genome-independent:

- TransABySS ('10), Trinity ('11), Oases ('12)

Genome-guided:

- Annotation-free:
  Scripture ('10), TRIP ('12), **Traph** (**T**ranscripts in G**raph**s)
- Annotation-guided:
  Cufflinks ('10), IsoLasso ('11), SLIDE ('11), iReckon ('12), CLIIQ ('12)

Graph models and main existing solutions

1. Overlap graph (Cufflinks)
   - each read is a node
   - if two reads overlap we add an edge between them

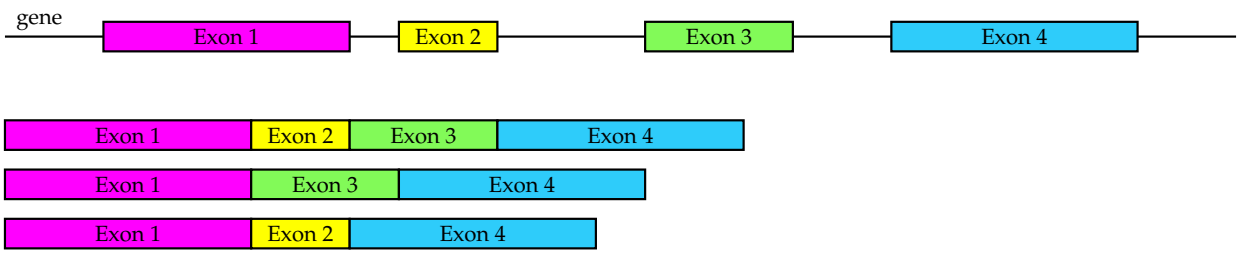
Graph models and main existing solutions

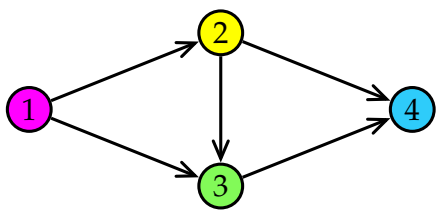
1. Overlap graph (Cufflinks)
   - each read is a node
   - if two reads overlap we add an edge between them
   - look for a path cover of minimum cost
   - estimate the expression levels of the paths in the cover

Graph models and main existing solutions

1. Overlap graph (Cufflinks)
   - each read is a node
   - if two reads overlap we add an edge between them
   - look for a path cover of minimum cost
   - estimate the expression levels of the paths in the cover

2. Splicing graph (most of the other tools)
   - detect exon boundaries from the spliced alignments
   - every node stands for an exon
   - every edge stands for reads spanning two consecutive exons

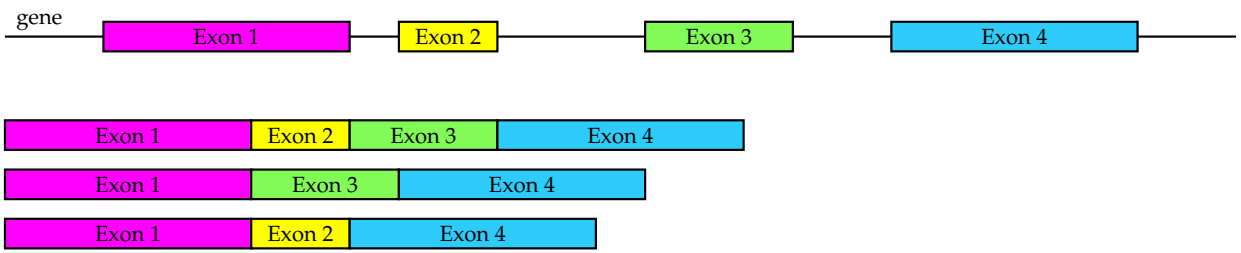


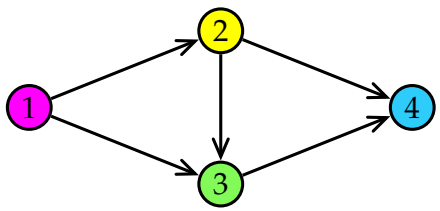


Graph models and main existing solutions

1. Overlap graph (Cufflinks)
   - each read is a node
   - if two reads overlap we add an edge between them
   - look for a path cover of minimum cost
   - estimate the expression levels of the paths in the cover

2. Splicing graph (most of the other tools)
   - detect exon boundaries from the spliced alignments
   - every node stands for an exon
   - every edge stands for reads spanning two consecutive exons



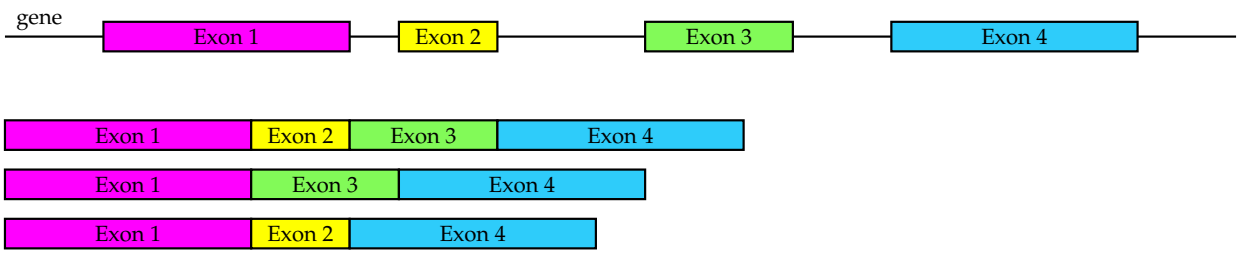


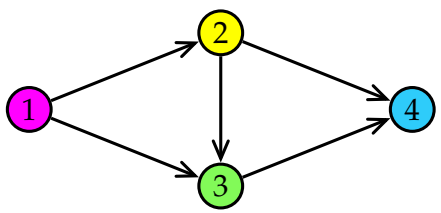
   - the splicing graph is a DAG
   - nodes and edges have observed coverages

Graph models and main existing solutions

1. Overlap graph (Cufflinks)
   - each read is a node
   - if two reads overlap we add an edge between them
   - look for a path cover of minimum cost
   - estimate the expression levels of the paths in the cover

2. Splicing graph (most of the other tools)
   - detect exon boundaries from the spliced alignments
   - every node stands for an exon
   - every edge stands for reads spanning two consecutive exons





   - the splicing graph is a DAG
   - nodes and edges have observed coverages
   - exhaustively enumerate all possible paths
   - choose the most likely ones based on their coverage using an ILP, QP, QP + LASSO, statistical methods

A unified problem formulation [RECOMB-Seq 2013]

Problem (Unannotated Transcript Expression Cover)

INPUT: *a splicing DAG* $G = (V, E)$*, and for all* $v \in V$ *and* $(u, v) \in E$*,*

- *observed coverage values* $cov(v)$ *and* $cov(u, v)$*, and*

A unified problem formulation [RECOMB-Seq 2013]

Problem (Unannotated Transcript Expression Cover)

INPUT: *a splicing DAG* $G = (V, E)$*, and for all* $v \in V$ *and* $(u, v) \in E$*,*

- *observed coverage values* $cov(v)$ *and* $cov(u, v)$*, and*
- *penalty functions* $f_v(\cdot)$ *and* $f_{uv}(\cdot)$

A unified problem formulation [RECOMB-Seq 2013]

Problem (Unannotated Transcript Expression Cover)

INPUT: *a splicing DAG $G = (V, E)$, and for all $v \in V$ and $(u, v) \in E$,*

- *observed coverage values $cov(v)$ and $cov(u, v)$, and*
- *penalty functions $f_v(\cdot)$ and $f_{uv}(\cdot)$*

FIND:

- *a tuple $\mathcal{P}$ of paths from the sources of G to the sinks of G,*

A unified problem formulation [RECOMB-Seq 2013]

Problem (Unannotated Transcript Expression Cover)

INPUT: *a splicing DAG* $G = (V, E)$*, and for all* $v \in V$ *and* $(u, v) \in E$*,*

- *observed coverage values* $cov(v)$ *and* $cov(u, v)$*, and*
- *penalty functions* $f_v(\cdot)$ *and* $f_{uv}(\cdot)$

FIND:

- *a tuple* $\mathcal{P}$ *of paths from the sources of* G *to the sinks of* G*,*
- *an expression level* $e(P)$ *for each path* $P \in \mathcal{P}$*,*

A unified problem formulation [RECOMB-Seq 2013]

Problem (Unannotated Transcript Expression Cover)

INPUT: *a splicing DAG $G = (V, E)$, and for all $v \in V$ and $(u, v) \in E$,*

- *observed coverage values $cov(v)$ and $cov(u, v)$, and*
- *penalty functions $f_v(\cdot)$ and $f_{uv}(\cdot)$*

FIND:

- *a tuple $\mathcal{P}$ of paths from the sources of G to the sinks of G,*
- *an expression level $e(P)$ for each path $P \in \mathcal{P}$,*

which minimize

$$\sum_{v \in V} f_v \left(\left| cov(v) - \sum_{P \in \mathcal{P}:\, v \in P} e(P) \right| \right) + \sum_{(u,v) \in E} f_{uv} \left(\left| cov(u, v) - \sum_{P \in \mathcal{P}:\, (u,v) \in P} e(P) \right| \right)$$

A unified problem formulation [RECOMB-Seq 2013]

Problem (Unannotated Transcript Expression Cover)

INPUT: *a splicing DAG* $G = (V, E)$*, and for all* $v \in V$ *and* $(u, v) \in E$*,*

- *observed coverage values* $cov(v)$ *and* $cov(u, v)$*, and*
- *penalty functions* $f_v(\cdot)$ *and* $f_{uv}(\cdot)$

FIND:

- *a tuple* $\mathcal{P}$ *of paths from the sources of* G *to the sinks of* G*,*
- *an expression level* $e(P)$ *for each path* $P \in \mathcal{P}$*,*

which minimize

$$\sum_{v \in V} f_v \left(\left| cov(v) - \sum_{P \in \mathcal{P}:\, v \in P} e(P) \right| \right) + \sum_{(u,v) \in E} f_{uv} \left(\left| cov(u,v) - \sum_{P \in \mathcal{P}:\, (u,v) \in P} e(P) \right| \right)$$

For example, if for all nodes v and edges (u, v),

- $f_v(x) = x, f_{uv}(x) = x \Rightarrow$ least sum of absolute differences model [CLIIQ]
- $f_u(x) = x^2, f_{uv}(x) = x^2 \Rightarrow$ least sum of squares model [IsoLasso, SLIDE]

A UNIFIED PROBLEM FORMULATION

- The problem is polynomially-time solvable by min-cost flows if the penalty functions are convex
- But in practice we are interested in parsimonious solutions

A unified problem formulation

- The problem is polynomially-time solvable by min-cost flows if the penalty functions are convex
- But in practice we are interested in parsimonious solutions

Problem (k-Unannotated Transcript Expression Cover)

INPUT: *same as before, and*

- *number of paths k*

A unified problem formulation

- The problem is polynomially-time solvable by min-cost flows if the penalty functions are convex
- But in practice we are interested in parsimonious solutions

Problem (k-Unannotated Transcript Expression Cover)

INPUT: *same as before, and*

- *number of paths k*

FIND:

- *a tuple $\mathcal{P}$ of k paths from the sources of G to the sinks of G,*

A unified problem formulation

- The problem is polynomially-time solvable by min-cost flows if the penalty functions are convex
- But in practice we are interested in parsimonious solutions

Problem (k-Unannotated Transcript Expression Cover)

INPUT: *same as before, and*

- *number of paths k*

FIND:

- *a tuple $\mathcal{P}$ of k paths from the sources of G to the sinks of G,*
- *an expression level $e(P)$ for each path $P \in \mathcal{P}$,*

A unified problem formulation

- The problem is polynomially-time solvable by min-cost flows if the penalty functions are convex
- But in practice we are interested in parsimonious solutions

Problem (k-Unannotated Transcript Expression Cover)

INPUT: *same as before, and*

- *number of paths k*

FIND:

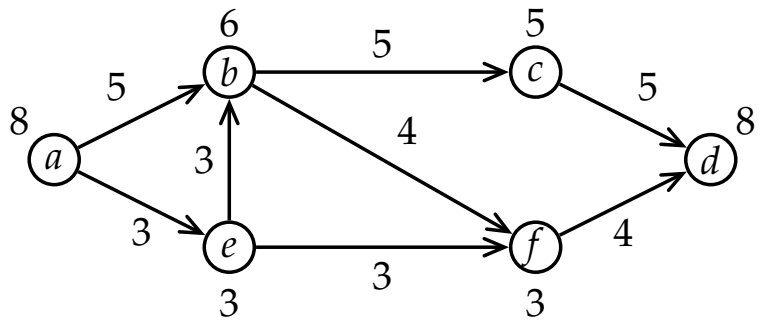
- *a tuple $\mathcal{P}$ of k paths from the sources of G to the sinks of G,*
- *an expression level $e(P)$ for each path $P \in \mathcal{P}$,*

which minimize

$$\sum_{v \in V} f_v \left(\left| cov(v) - \sum_{P \in \mathcal{P}:\, v \in P} e(P) \right| \right) + \sum_{(u,v) \in E} f_{uv} \left(\left| cov(u,v) - \sum_{P \in \mathcal{P}:\, (u,v) \in P} e(P) \right| \right)$$

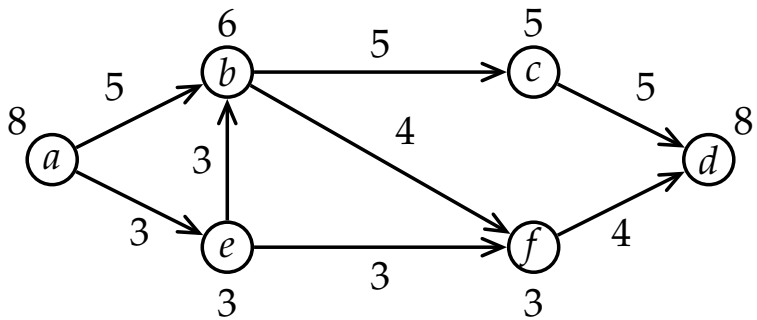
EXAMPLE $f_v(x) = x^2, f_{uv}(x) = x^2, k = 2$

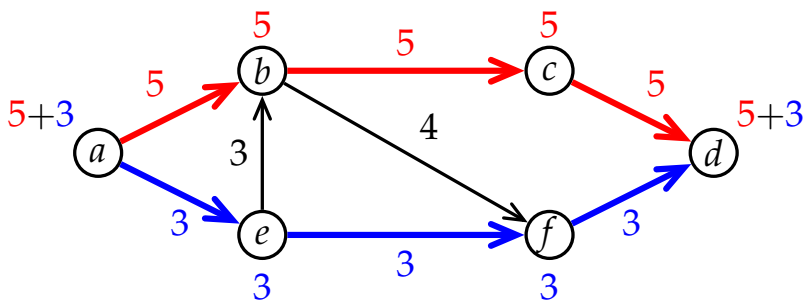
$$\sum_{v \in V} \left(cov(v) - \sum_{P \in \mathcal{P} : v \in P} e(P) \right)^2 + \sum_{(u,v) \in E} \left(cov(u, v) - \sum_{P \in \mathcal{P} : (u,v) \in P} e(P) \right)^2$$



EXAMPLE $f_v(x) = x^2, f_{uv}(x) = x^2, k = 2$

$$\sum_{v \in V} \left(cov(v) - \sum_{P \in \mathcal{P}:\, v \in P} e(P) \right)^2 + \sum_{(u,v) \in E} \left(cov(u,v) - \sum_{P \in \mathcal{P}:\, (u,v) \in P} e(P) \right)^2$$

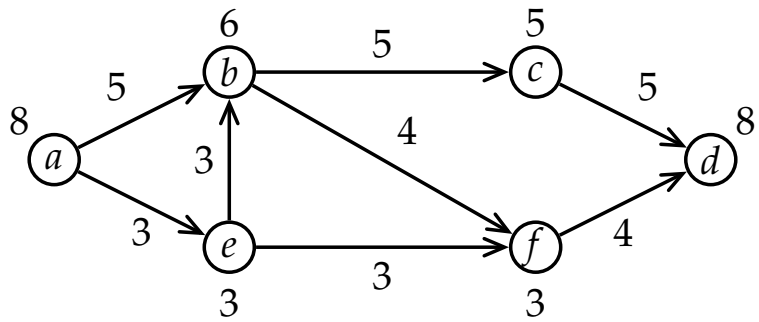


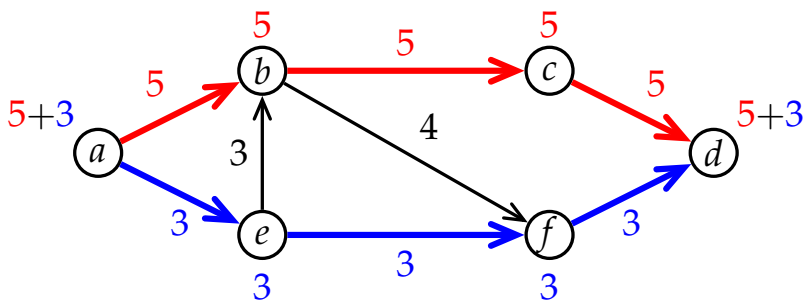


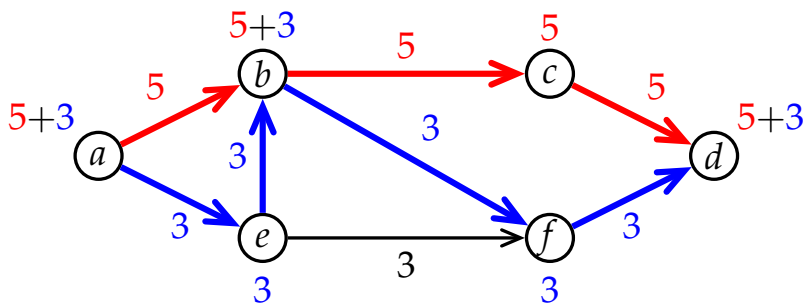
- [Left] A non-optimal tuple of 2 paths with cost $1 + 1 + 3^3 + 4^2 = 27$, from b, (f, d), (e, b), (b, f)

EXAMPLE $f_v(x) = x^2, f_{uv}(x) = x^2, k = 2$

$$\sum_{v \in V} \left(cov(v) - \sum_{P \in \mathcal{P} : v \in P} e(P) \right)^2 + \sum_{(u,v) \in E} \left(cov(u, v) - \sum_{P \in \mathcal{P} : (u,v) \in P} e(P) \right)^2$$







- [Left] A non-optimal tuple of 2 paths with cost $1 + 1 + 3^3 + 4^2 = 27$, from b, (f, d), (e, b), (b, f)
- [Right] The optimal tuple of 2 paths with cost $2^2 + 1 + 1 + 3^2 = 15$, from b, and (b, f), (f, d), (e, f)

Computational Complexity

Theorem

If the penalty functions f_v and f_{uv} are such that $f_v(0) = 0$, $f_{uv}(0) = 0$, and $f_v(x) > 0$, $f_{uv}(x) > 0$ for all $x > 0$, then Problem k-UTEC is NP-hard in the strong sense.

Computational Complexity

Theorem

*If the penalty functions f_v and f_{uv} are such that $f_v(0) = 0$, $f_{uv}(0) = 0$, and $f_v(x) > 0$, $f_{uv}(x) > 0$ for all $x > 0$, then Problem k-**UTEC** is NP-hard in the strong sense.*

- The proof reduces from the 3-PARTITION Problem
- Our proof idea was already employed by [Li, Jiang, Zhang, arXiv, 2013] to show that the Isoform Reconstruction by Maximum Likelihood Problem, deployed in tools such as iReckon, NSMAP, Montebello, is also NP-hard

A dynamic programming algorithm

We can exploit the fact that the input splicing graph G is acyclic.

- Fix a k-tuple of expression levels
- For each k-tuple of vertices of G, store the cost of the optimal paths ending in these k vertices

A dynamic programming algorithm

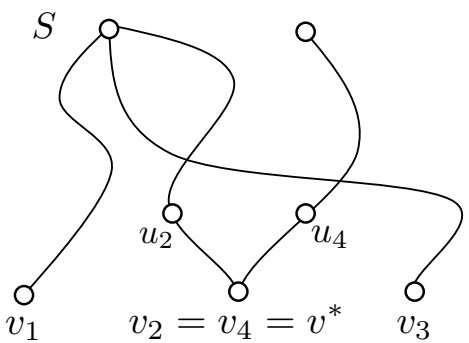
We can exploit the fact that the input splicing graph G is acyclic.

- Fix a k-tuple of expression levels
- For each k-tuple of vertices of G, store the cost of the optimal paths ending in these k vertices
- Compute the cost of a k-tuple from the cost of all k-tuples immediately preceding it

A DYNAMIC PROGRAMMING ALGORITHM

We can exploit the fact that the input splicing graph G is acyclic.

- Fix a k-tuple of expression levels
- For each k-tuple of vertices of G, store the cost of the optimal paths ending in these k vertices
- Compute the cost of a k-tuple from the cost of all k-tuples immediately preceding it

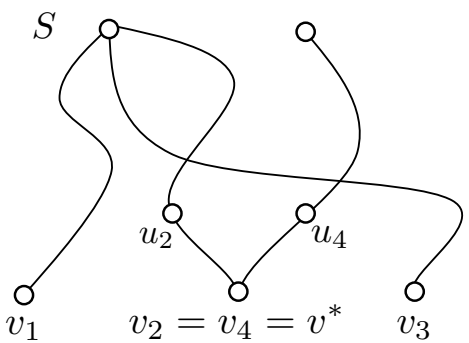


E.g., (v_1, v_2, v_3, v_4) with $v_2 = v_4 = v^*$ is immediately preceded by (v_1, u_2, v_2, u_4)

A dynamic programming algorithm

We can exploit the fact that the input splicing graph G is acyclic.

- Fix a k-tuple of expression levels
- For each k-tuple of vertices of G, store the cost of the optimal paths ending in these k vertices
- Compute the cost of a k-tuple from the cost of all k-tuples immediately preceding it



E.g., (v_1, v_2, v_3, v_4) with $v_2 = v_4 = v^*$ is immediately preceded by (v_1, u_2, v_2, u_4)

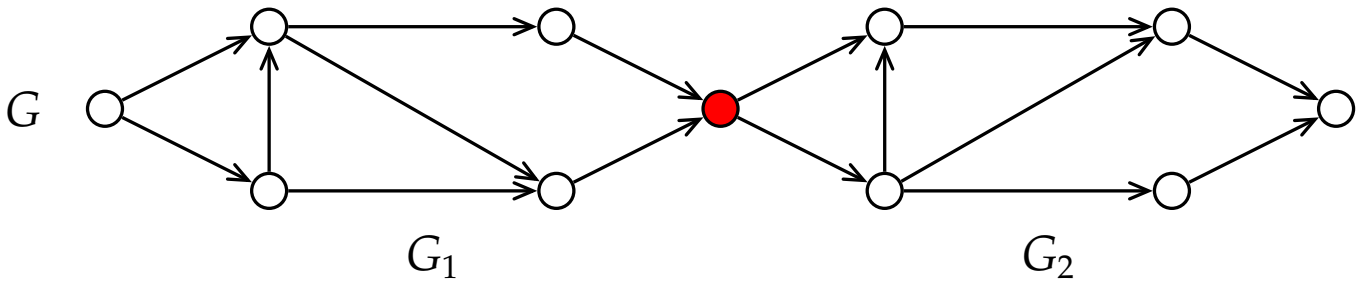
Theorem

*If the penalty functions positive-valued then Problem k-**UTEC** can be solved in time* $O(|M|^k \Delta^k n^k)$,
where $n := |V(G)|$, *we assume that* M *is the set of possible expression levels, and the maximum in-degree of* G *is* Δ.

Heuristics and optimizations

For a practical implementation, we

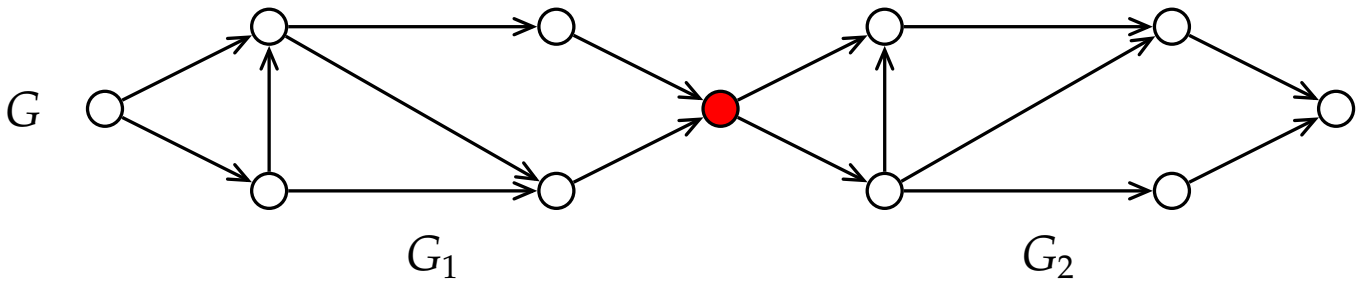
1. Decompose the problem along cut nodes



Heuristics and optimizations

For a practical implementation, we

1. Decompose the problem along cut nodes

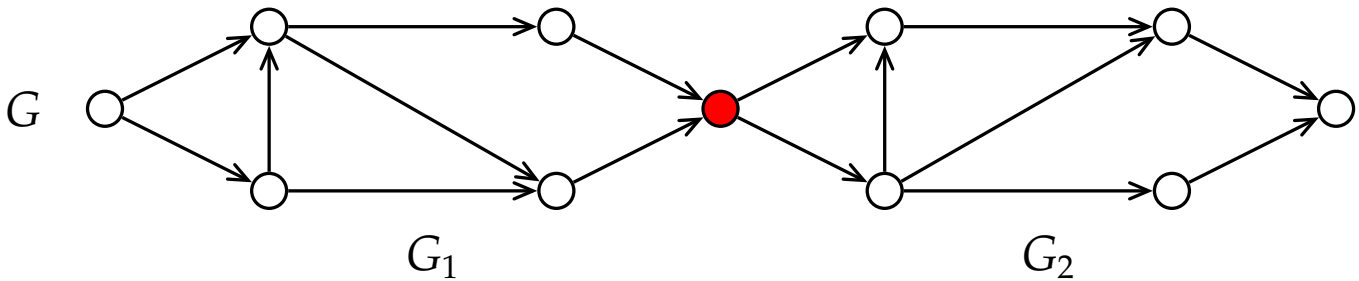


2. Employ a genetic algorithm for finding the optimal expression levels

Heuristics and optimizations

For a practical implementation, we

1. Decompose the problem along cut nodes

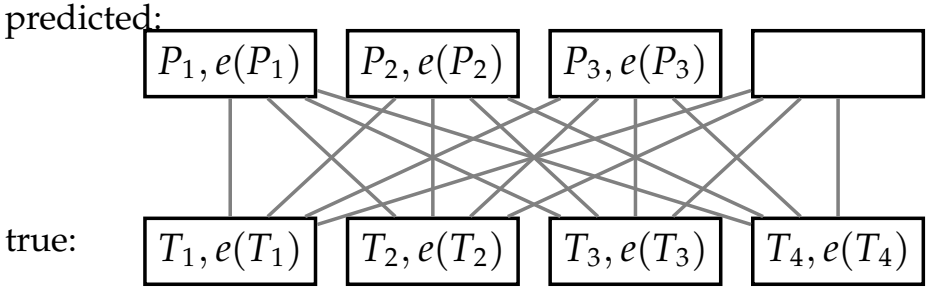


2. Employ a genetic algorithm for finding the optimal expression levels

3. Reduce the exponential dependency on k by iteratively looking for the optimal $k' < k$ paths and removing their coverage from the graph, until obtaining k paths

VALIDATION

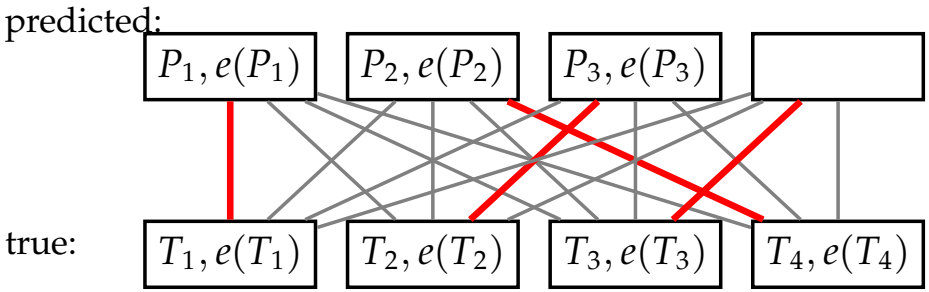
- construct a bipartite graph with predicted and true transcripts



- the edge weight between $[P_i, e(P_i)]$ and $[T_j, e(T_j)]$ is a combined measure of
  - *sequence dissimilarity* := $\frac{\text{edit distance between } T_j \text{ and } P_i}{\max(|T_j|, |P_i|)}$
  - *relative expression level difference* := $\frac{|e(T_j) - e(P_i)|}{e(T_j)}$

VALIDATION

- construct a bipartite graph with predicted and true transcripts

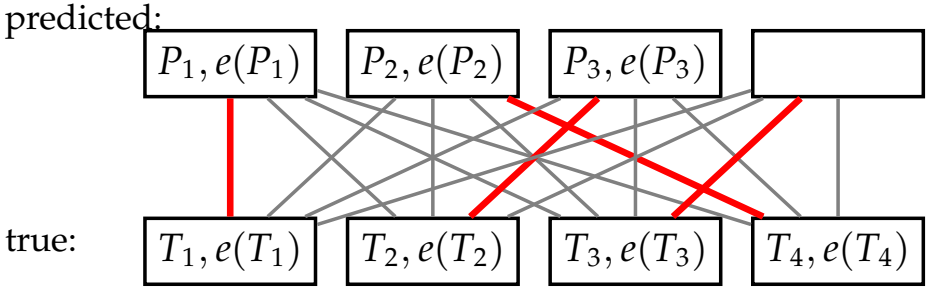


- the edge weight between $[P_i, e(P_i)]$ and $[T_j, e(T_j)]$ is a combined measure of
  - *sequence dissimilarity* := $\frac{\text{edit distance between } T_j \text{ and } P_i}{\max(|T_j|, |P_i|)}$
  - *relative expression level difference* := $\frac{|e(T_j) - e(P_i)|}{e(T_j)}$

- compute minimum weight perfect matching

Validation

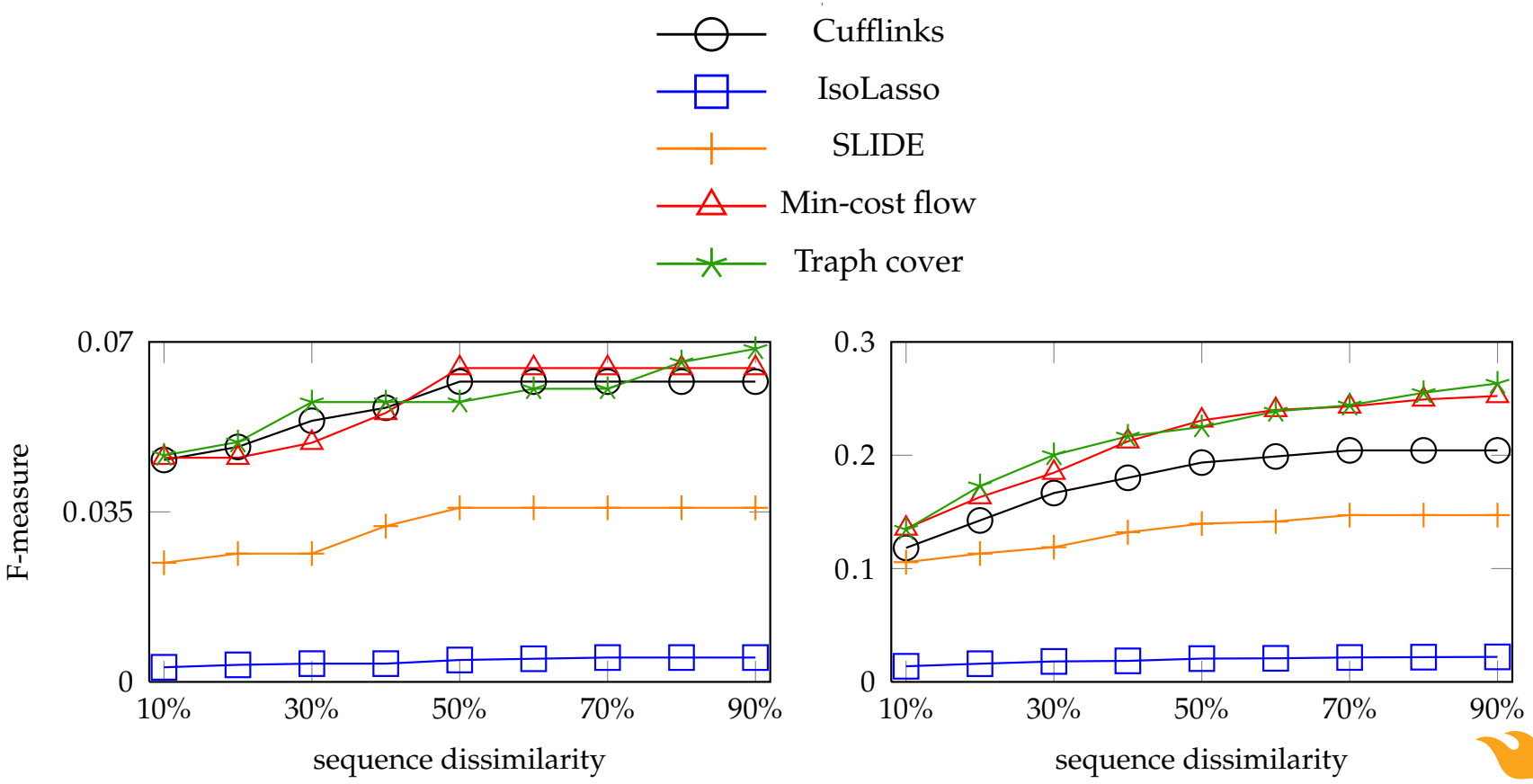
- construct a bipartite graph with predicted and true transcripts



- the edge weight between $[P_i, e(P_i)]$ and $[T_j, e(T_j)]$ is a combined measure of
  - *sequence dissimilarity* := $\frac{\text{edit distance between } T_j \text{ and } P_i}{\max(|T_j|, |P_i|)}$
  - *relative expression level difference* := $\frac{|e(T_j) - e(P_i)|}{e(T_j)}$
- compute minimum weight perfect matching
- a True Positive is a match with sequence dissimilarity and expression difference under given thresholds
- other events define False Positives and False Negatives
- compute precision, recall, F-measure

Experimental results on simulated data

- Simulated paired-end reads from the transcripts of 1,462 genes in HC 2
- Reads aligned with TopHat
- Alignments for all genes combined into one file, fed to the tools



(a) expr. difference threshold 10%

(b) expr. difference threshold 90%

Experimental results on real data

- 2,406,339 paired-end reads of length 75bp mapping to HC 2
- 735 genes where all tools made predictions
- 6,325 annotated transcripts in total

Tool	Total predicted	Shared with annotation at sequence dissimilarity under				
		10%	20%	30%	40%	50%
Cufflinks	1916	648	955	1171	1307	1413
IsoLasso	1468	589	782	923	1022	1100
SLIDE	2229	635	983	1242	1391	1474
Min-cost flow	2148	722	1000	1228	1341	1456
Traph cover	2109	**788**	**1063**	**1283**	**1407**	**1501**

Conclusions cs.helsinki.fi/gsa/traph/

- A unified problem formulation for transcript identification and quantification
- We replace the exhaustive enumeration of all (tuples of) paths by enumeration of all k-tuples of vertices
- We can increase the accuracy of the min-cost flow solution by tackling an NP-hard problem

Conclusions cs.helsinki.fi/gsa/traph/

- A unified problem formulation for transcript identification and quantification
- We replace the exhaustive enumeration of all (tuples of) paths by enumeration of all k-tuples of vertices
- We can increase the accuracy of the min-cost flow solution by tackling an NP-hard problem

Future work:

- integrate paired-end information
- procure real ground-truth
- exploit graph-width measures (e.g. tree-width), write approximation algorithms
- apply to other multi-assembly problems