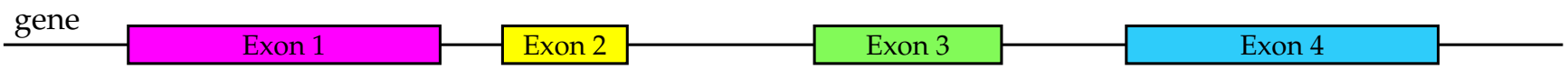
A Novel Min-Cost Flow Method for Estimating Transcript Expression with RNA-Seq

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

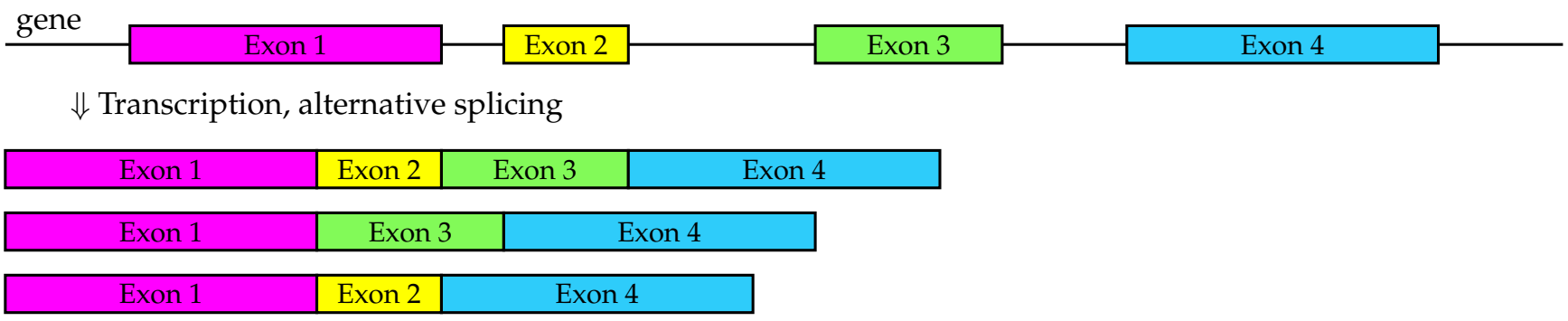
[1]*Helsinki Institute for Information Technology HIIT, Department of Computer Science, University of Helsinki, Finland*
[2]*Department of Computer Science, University of Verona, Italy*

RECOMB-Seq
April 11, 2013

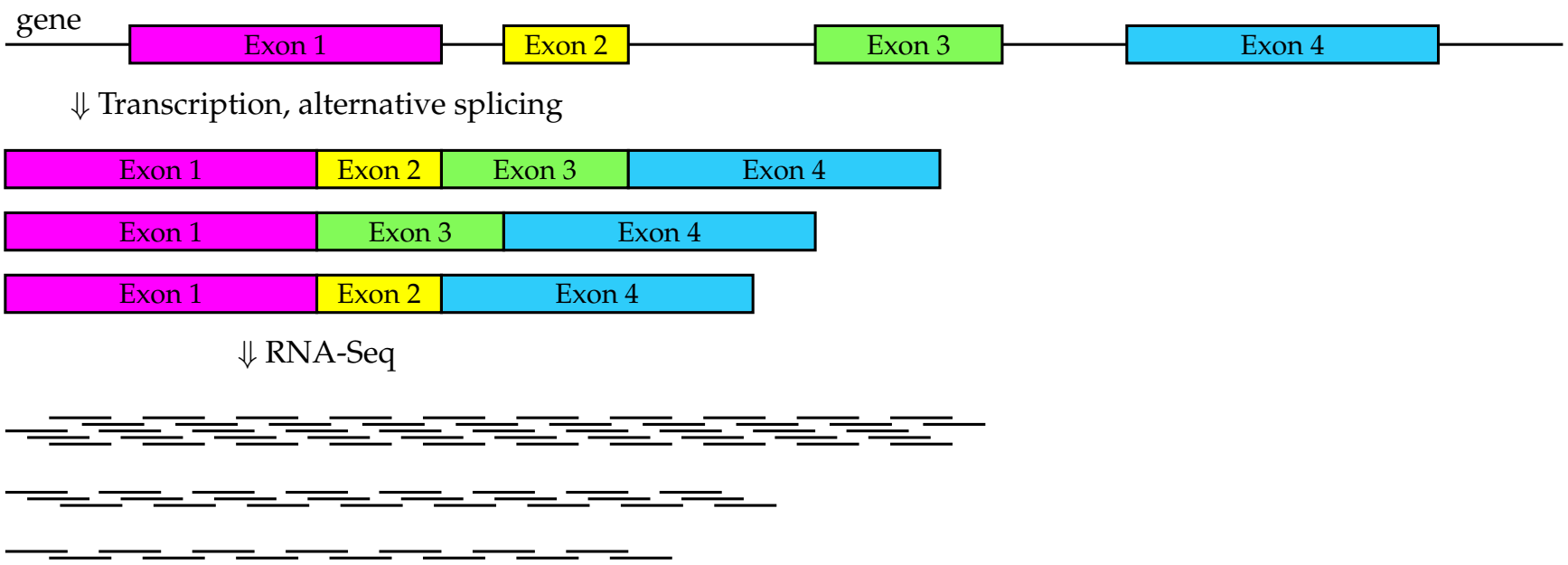
THE BIOLOGICAL PROBLEM



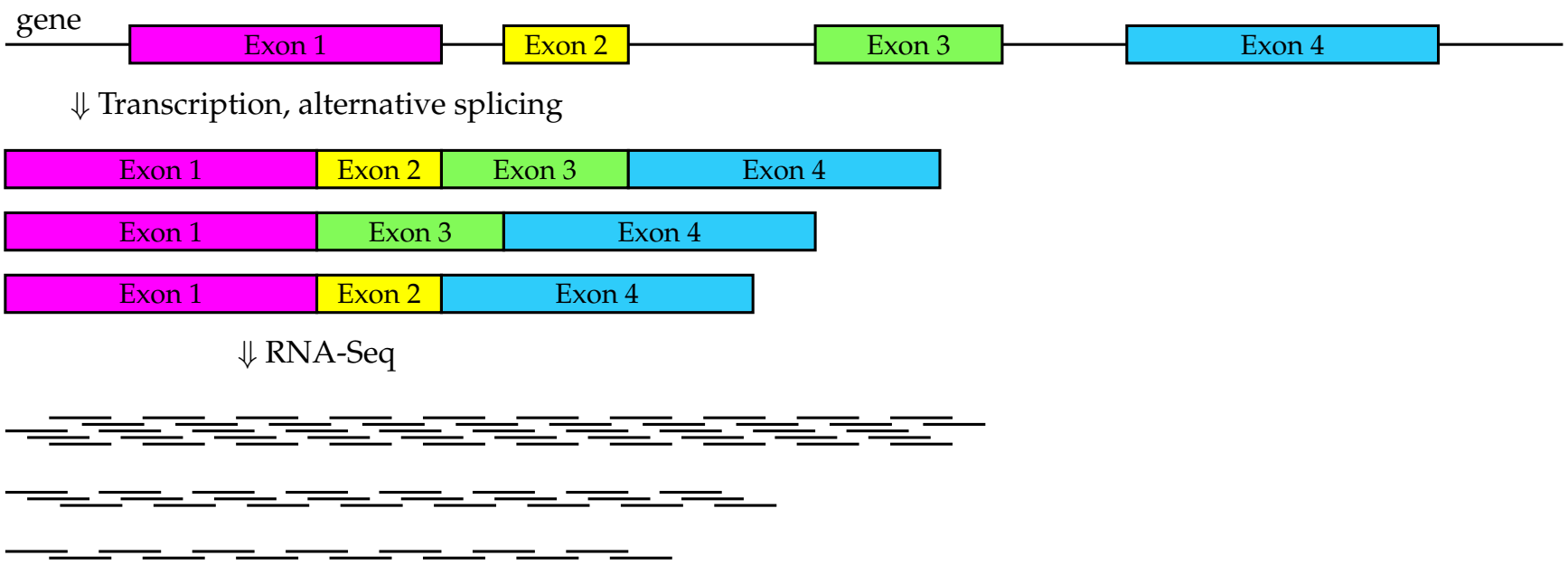
THE BIOLOGICAL PROBLEM



The biological problem



THE BIOLOGICAL PROBLEM



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

Existing methods

Genome-independent:

- AbySS ('09)

Existing methods

Genome-independent:

- AbySS ('09)

Genome-guided:

- Spliced alignment to the genome:
  TopHat ('09), SpliceMap ('10), GMAP ('05-'10),...

Existing methods

Genome-independent:

- AbySS ('09)

Genome-guided:

- Spliced alignment to the genome:
  TopHat ('09), SpliceMap ('10), GMAP ('05-'10),...
- Annotation-free:
  Scripture ('10), TRIP ('12),

Existing methods

Genome-independent:

- AbySS ('09)

Genome-guided:

- Spliced alignment to the genome:
  TopHat ('09), SpliceMap ('10), GMAP ('05-'10),...
- Annotation-free:
  Scripture ('10), TRIP ('12),
- Annotation-guided:
  Cufflinks ('10), IsoLasso ('11), SLIDE ('11), iReckon ('12), CLIIQ ('12)

Existing methods

Genome-independent:

- AbySS ('09)

Genome-guided:

- Spliced alignment to the genome:
  TopHat ('09), SpliceMap ('10), GMAP ('05-'10),...
- 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

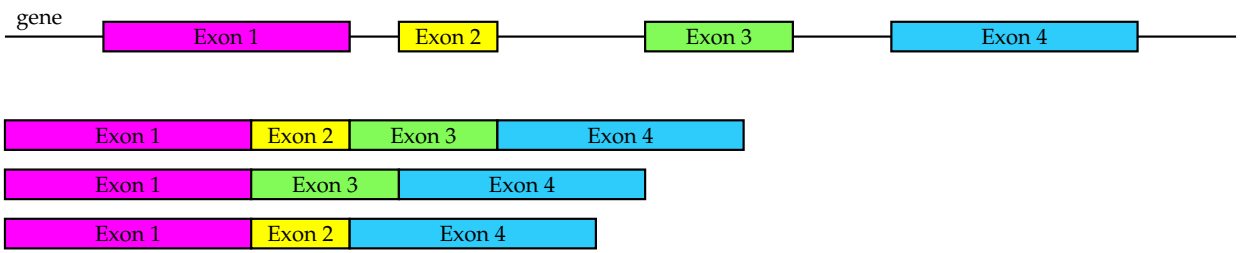
Graph models and main existing solutions

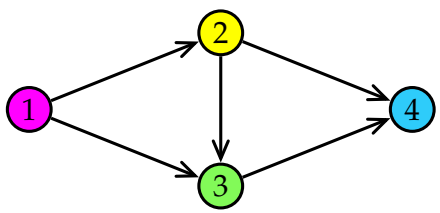
1. Overlap graph (Cufflinks)
   - each read is a node
   - if two reads overlap we add an edge between them
   - we look for a path cover of minimum cost
   - estimate the expression levels of the paths in the cover

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
   - we look for a path cover of minimum cost
   - estimate the expression levels of the paths in the cover

2. Splicing graph (almost all 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

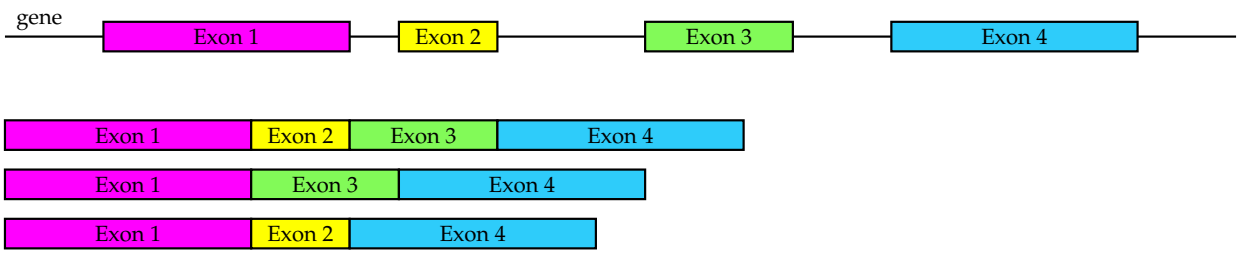


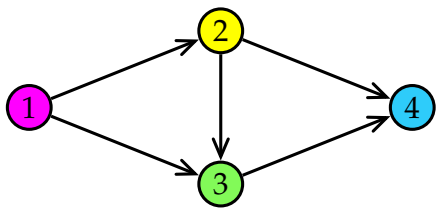


Graph models and main existing solutions

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

2. Splicing graph (almost all 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



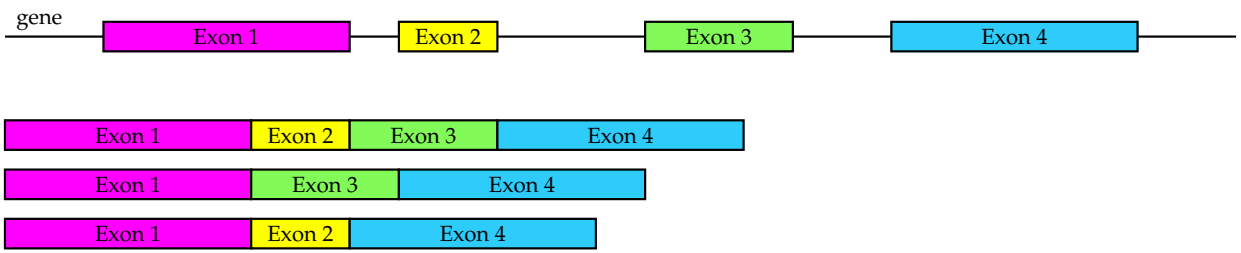


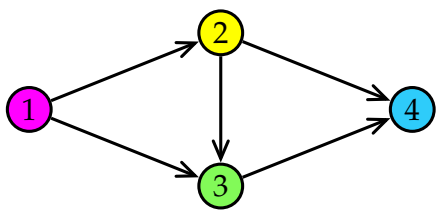
   - the splicing graph is a DAG
   - nodes and edges have observed coverages

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
   - we look for a path cover of minimum cost
   - estimate the expression levels of the paths in the cover

2. Splicing graph (almost all 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

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

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*
- *cost functions* $f_v(\cdot)$ *and* $f_{uv}(\cdot)$

A unified problem formulation

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*
- *cost 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

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*
- *cost 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,*
- *with an estimated expression level $e(P)$ for each path $P \in \mathcal{P}$,*

A unified problem formulation

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*
- *cost 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,*
- *with an estimated 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

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*
- *cost 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,*
- *with an estimated expression level $e(P)$ for each path $P \in \mathcal{P}$,*

which minimize

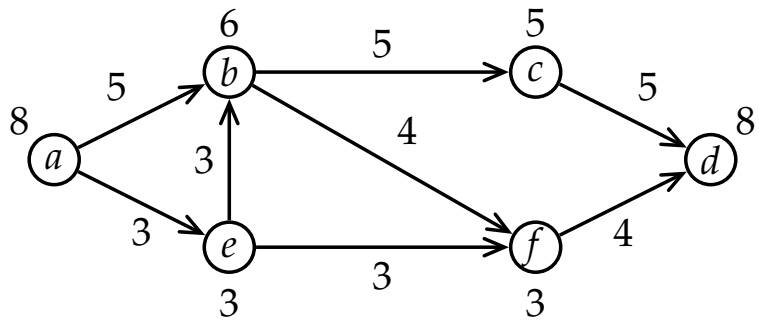
$$\sum_{v \in V} f_v \left(\left| cov(v) - \sum_{P \in \mathcal{P}:\, v \in P} e(P) \right| \right) + \sum_{(u,v) \in E} f_{uv} \left(\left| cov(u, v) - \sum_{P \in \mathcal{P}:\, (u,v) \in P} e(P) \right| \right)$$

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]

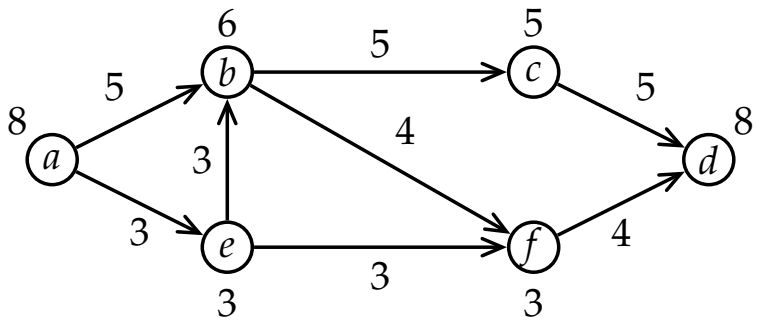
Example $f_v(x) = x^2, f_{uv}(x) = x^2$

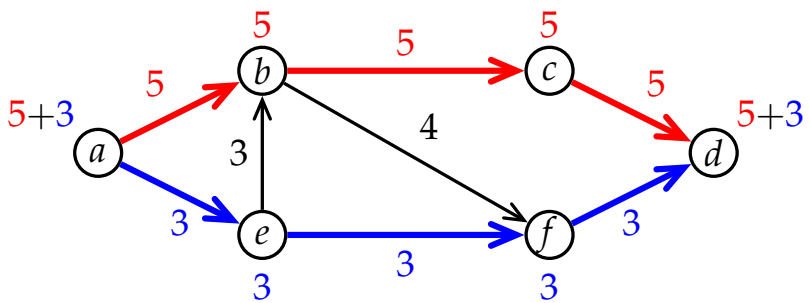
$$\sum_{v \in V} \left(cov(v) - \sum_{P \in \mathcal{P} : v \in P} e(P) \right)^2 + \sum_{(u,v) \in E} \left(cov(u, v) - \sum_{P \in \mathcal{P} : (u,v) \in P} e(P) \right)^2$$



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

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

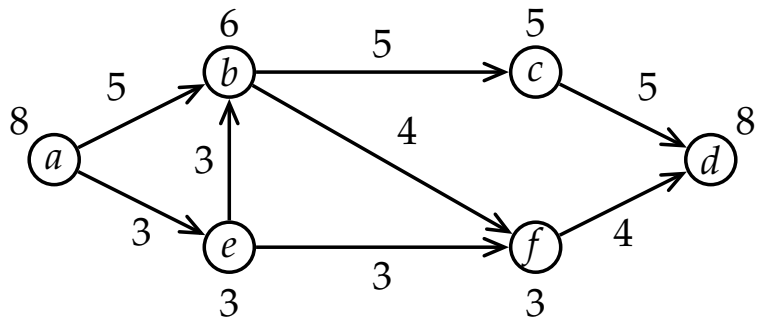


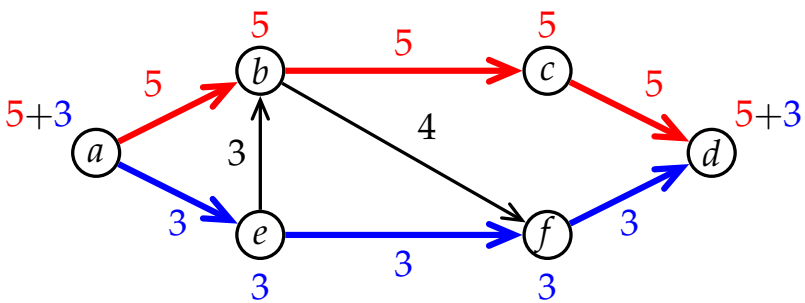


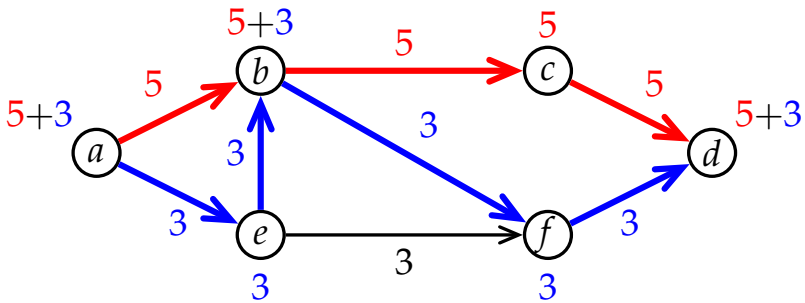
- [Left] A non-optimal tuple of paths with cost $1 + 1 + 3^3 + 4^2 = 27$, from b, (f, d), (e, b), (b, f)

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

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







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

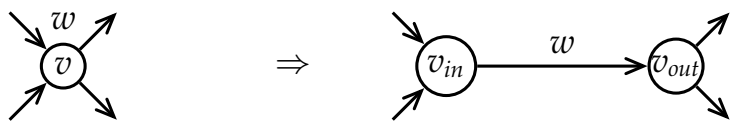
SOLUTION: REDUCTION TO NETWORK FLOWS

- A *flow* over a *flow network* $N = (V, E, b)$ is a function x assigning to every arc $(u, v) \in E$ a number $x_{uv} \in \mathbb{N}$ such that
  1. $0 \leq x_{uv} \leq b_{uv}$, for every $(u, v) \in E$,
  2. $\sum_{u \in V} x_{vu} = \sum_{u \in V} x_{uv}$, for every $v \in V$ not a source or a sink
  3. the flow exiting the sources = the flow entering the sinks (:= *value of the flow*)

Solution: reduction to network flows

- A *flow* over a *flow network* $N = (V, E, b)$ is a function x assigning to every arc $(u, v) \in E$ a number $x_{uv} \in \mathbb{N}$ such that
  1. $0 \leq x_{uv} \leq b_{uv}$, for every $(u, v) \in E$,
  2. $\sum_{u \in V} x_{vu} = \sum_{u \in V} x_{uv}$, for every $v \in V$ not a source or a sink
  3. the flow exiting the sources = the flow entering the sinks (:= *value of the flow*)

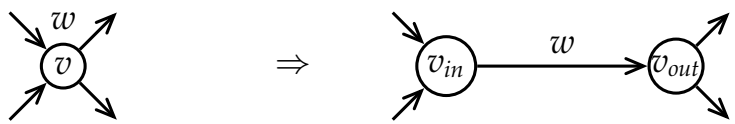
- We first reduce the problem to one having coverages only on edges



Solution: reduction to network flows

- A *flow* over a *flow network* $N = (V, E, b)$ is a function x assigning to every arc $(u, v) \in E$ a number $x_{uv} \in \mathbb{N}$ such that
  1. $0 \leq x_{uv} \leq b_{uv}$, for every $(u, v) \in E$,
  2. $\sum_{u \in V} x_{vu} = \sum_{u \in V} x_{uv}$, for every $v \in V$ not a source or a sink
  3. the flow exiting the sources = the flow entering the sinks (:= *value of the flow*)

- We first reduce the problem to one having coverages only on edges

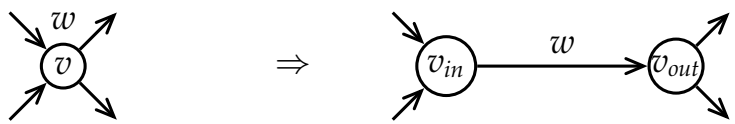


- Any collection of weighted paths in a graph induces a flow, and viceversa, any flow can be split into (linearly many) paths

Solution: reduction to network flows

- A *flow* over a *flow network* $N = (V, E, b)$ is a function x assigning to every arc $(u, v) \in E$ a number $x_{uv} \in \mathbb{N}$ such that
  1. $0 \leq x_{uv} \leq b_{uv}$, for every $(u, v) \in E$,
  2. $\sum_{u \in V} x_{vu} = \sum_{u \in V} x_{uv}$, for every $v \in V$ not a source or a sink
  3. the flow exiting the sources = the flow entering the sinks (:= *value of the flow*)

- We first reduce the problem to one having coverages only on edges



- Any collection of weighted paths in a graph induces a flow, and viceversa, any flow can be split into (linearly many) paths

- We will find an "optimal" flow: the value of the flow on each edge will be the predicted coverage on that edge

SOLUTION: MIN-COST FLOWS

- In a min-cost flow problem, one is additionally given
  - a flow value to be pushed from sources to sinks, and
  - flow cost functions $c_{uv}(\cdot)$, for every arc $(u, v) \in E$

  and is required to find a flow of given value which minimizes:

$$\sum_{(u,v)\in E} c_{uv}(x_{uv})$$

SOLUTION: MIN-COST FLOWS

- In a min-cost flow problem, one is additionally given
  - a flow value to be pushed from sources to sinks, and
  - flow cost functions $c_{uv}(\cdot)$, for every arc $(u, v) \in E$

  and is required to find a flow of given value which minimizes:

$$\sum_{(u,v)\in E} c_{uv}(x_{uv})$$

  - If the cost functions are convex, then we can find a min-cost flow in polynomial time

Solution: min-cost flows

- In a min-cost flow problem, one is additionally given
  - a flow value to be pushed from sources to sinks, and
  - flow cost functions $c_{uv}(\cdot)$, for every arc $(u, v) \in E$

  and is required to find a flow of given value which minimizes:

$$\sum_{(u,v)\in E} c_{uv}(x_{uv})$$

- If the cost functions are convex, then we can find a min-cost flow in polynomial time
- We can further polynomially transform the splicing graph to a flow network modeling the offsets between observed and predicted coverage

Solution: min-cost flows

- In a min-cost flow problem, one is additionally given
  - a flow value to be pushed from sources to sinks, and
  - flow cost functions $c_{uv}(\cdot)$, for every arc $(u, v) \in E$

  and is required to find a flow of given value which minimizes:

$$\sum_{(u,v)\in E} c_{uv}(x_{uv})$$

  - If the cost functions are convex, then we can find a min-cost flow in polynomial time
  - We can further polynomially transform the splicing graph to a flow network modeling the offsets between observed and predicted coverage

Theorem

Given an input splicing DAG as input for the Problem UTEC, if the cost functions are convex, we can find the optimal coverage of each node and edge in polynomial time.

Solution: min-cost flows

- In a min-cost flow problem, one is additionally given
  - a flow value to be pushed from sources to sinks, and
  - flow cost functions $c_{uv}(\cdot)$, for every arc $(u, v) \in E$

  and is required to find a flow of given value which minimizes:

$$\sum_{(u,v)\in E} c_{uv}(x_{uv})$$

  - If the cost functions are convex, then we can find a min-cost flow in polynomial time
  - We can further polynomially transform the splicing graph to a flow network modeling the offsets between observed and predicted coverage

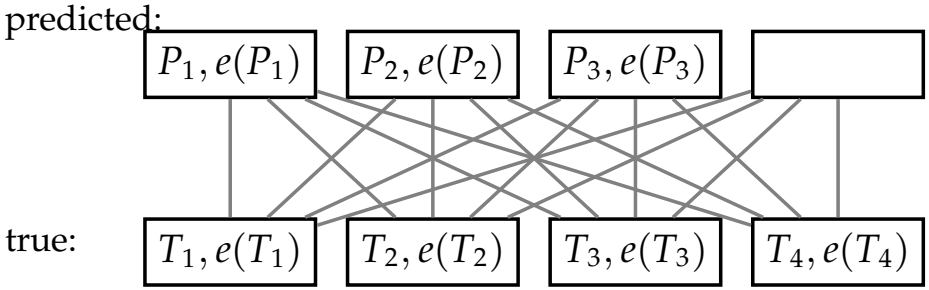
Theorem

Given an input splicing DAG as input for the Problem UTEC, if the cost functions are convex, we can find the optimal coverage of each node and edge in polynomial time.

- Splitting a flow into the minimum number of paths is NP-hard
- We currently apply a heuristic iteratively selecting the path of maximum bandwidth

Validation

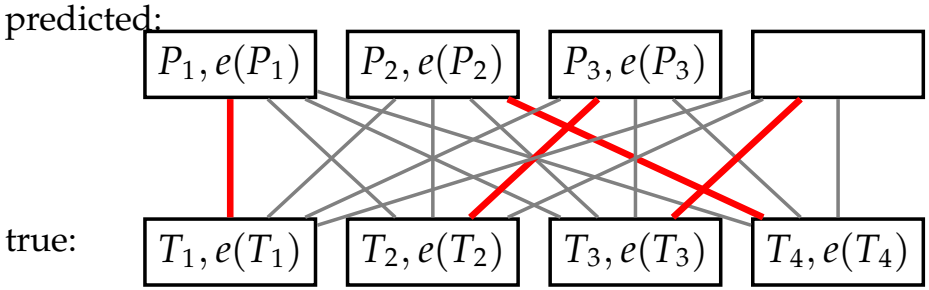
- construct a bipartite graph with predicted and true transcripts



- the edge weight between $P_i, e(P_i)$ and $T_j, e(T_j)$ is a combined measure between (cf. Normalized Compression Distance)
  - *bitscore* := $\frac{\text{\#bits needed to encode } T_j \text{ given } P_i}{\text{\#bits needed to encode } T_j \text{ just by inserts}}$
  - *relative expression level difference* := $\frac{|e(T_j) - e(P_i)|}{e(T_j)}$

VALIDATION

- construct a bipartite graph with predicted and true transcripts

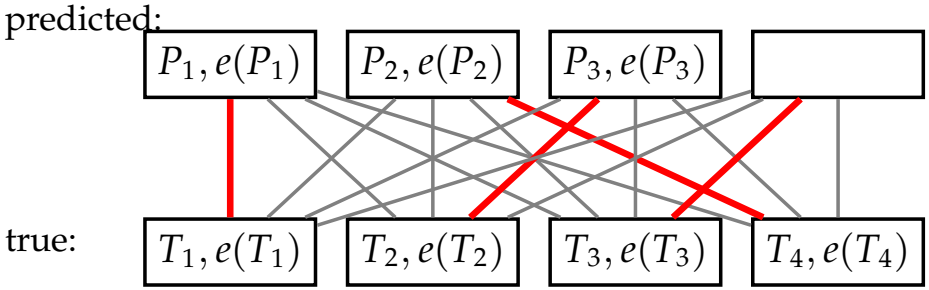


- the edge weight between $P_i, e(P_i)$ and $T_j, e(T_j)$ is a combined measure between (cf. Normalized Compression Distance)
  - *bitscore* := $\frac{\text{\#bits needed to encode } T_j \text{ given } P_i}{\text{\#bits needed to encode } T_j \text{ just by inserts}}$
  - *relative expression level difference* := $\frac{|e(T_j) - e(P_i)|}{e(T_j)}$

- 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 between (cf. Normalized Compression Distance)
  - *bitscore* := $\frac{\text{\#bits needed to encode } T_j \text{ given } P_i}{\text{\#bits needed to encode } T_j \text{ just by inserts}}$
  - *relative expression level difference* := $\frac{|e(T_j) - e(P_i)|}{e(T_j)}$

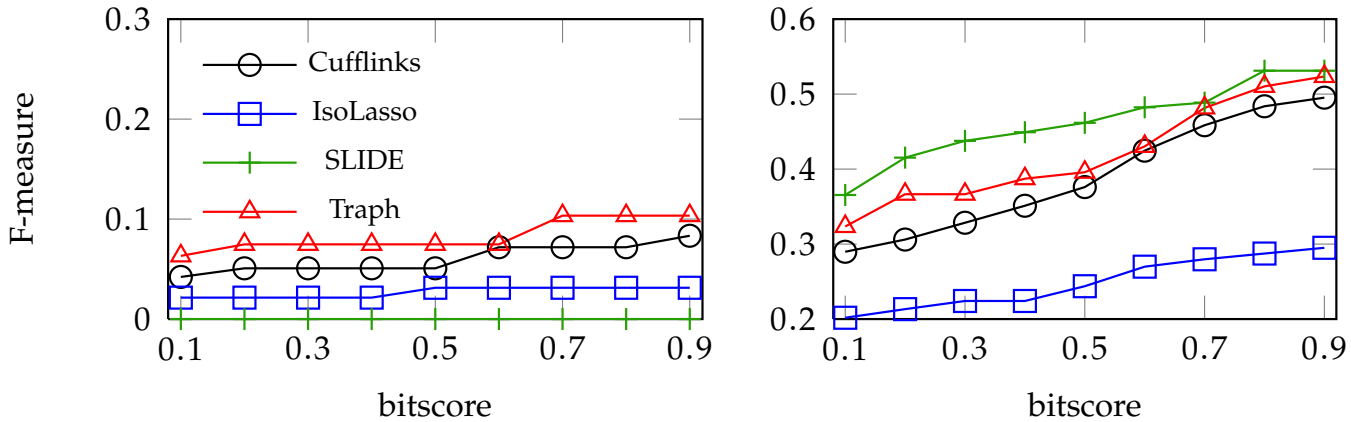
- compute minimum weight perfect matching

- a True Positive is a match with bitscore and expression difference under given thresholds
- other events define False Positives and False Negatives
- compute precision, recall, F-measure

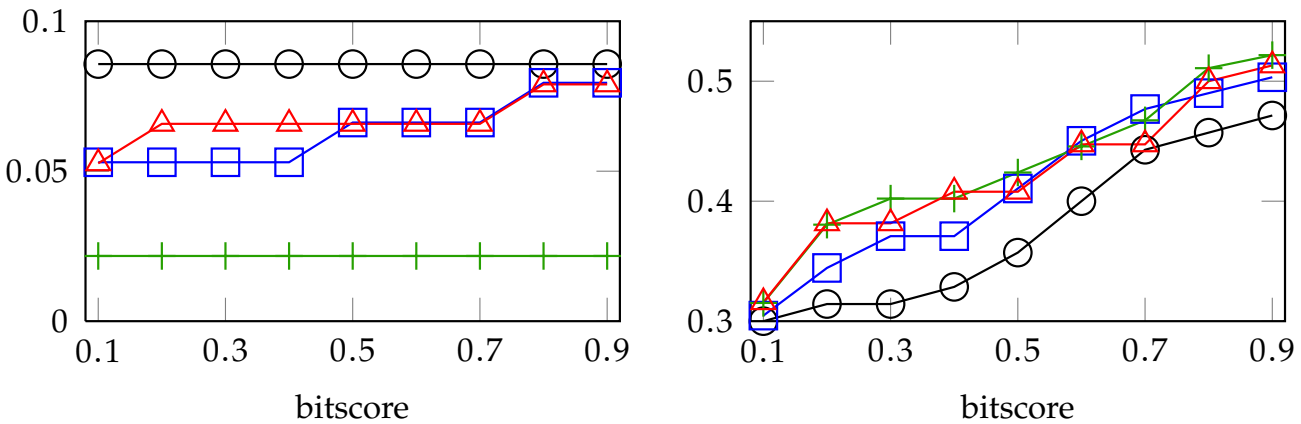
Experimental results on simulated data

- Simulated paired-end reads from the annotated transcripts of 29 genes
- Reads aligned with TopHat

1. Alignments fed to the tools for each gene independently



2. Alignments for all genes combined into one file, fed to the tools



(c) expr. difference threshold 0.1 (d) expr. difference threshold 0.9

Experimental results on real data

- 2 406 339 paired-end reads of length 75bp mapping to human chromosome 2
- 342 genes where all tools made predictions
- 3306 annotated transcripts in total

Tool	Total predicted	Shared with annotation, bitscore under				
		0.1	0.2	0.3	0.4	0.5
Cufflinks	933	128	200	267	342	410
IsoLasso	742	132	199	262	310	361
SLIDE	1131	191	281	381	463	552
Traph	961	157	255	330	398	474

Conclusions cs.helsinki.fi/gsa/traph/

- A novel unified problem formulation for transcript identification and quantification with RNA-Seq
- A polynomial-time solution by a reduction to min-cost network flows
- A general framework applicable to other multi-assembly problems

Conclusions cs.helsinki.fi/gsa/traph/

- A novel unified problem formulation for transcript identification and quantification with RNA-Seq
- A polynomial-time solution by a reduction to min-cost network flows
- A general framework applicable to other multi-assembly problems

Future work:

- integrate paired-end information
- integrate annotation information
- procure real ground-truth

Conclusions cs.helsinki.fi/gsa/traph/

- A novel unified problem formulation for transcript identification and quantification with RNA-Seq
- A polynomial-time solution by a reduction to min-cost network flows
- A general framework applicable to other multi-assembly problems

Future work:

- integrate paired-end information
- integrate annotation information
- procure real ground-truth
- if we limit by k the number of paths in an optimal solution
  - the problem becomes NP-hard
  - we can write dynamic programming algorithms which work well assuming k is not too large
  - we get better accuracy
  - $\Rightarrow$ see our poster