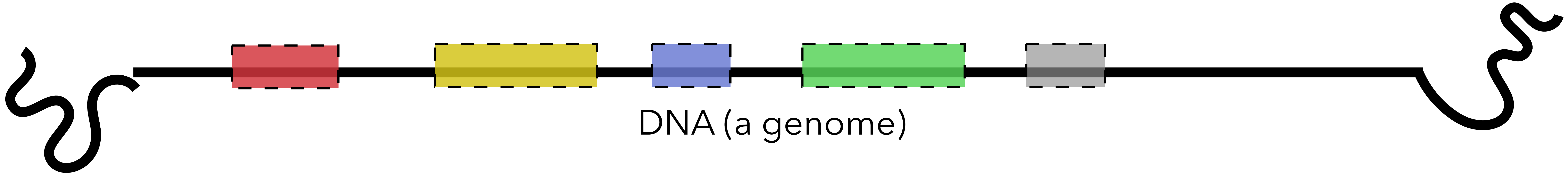
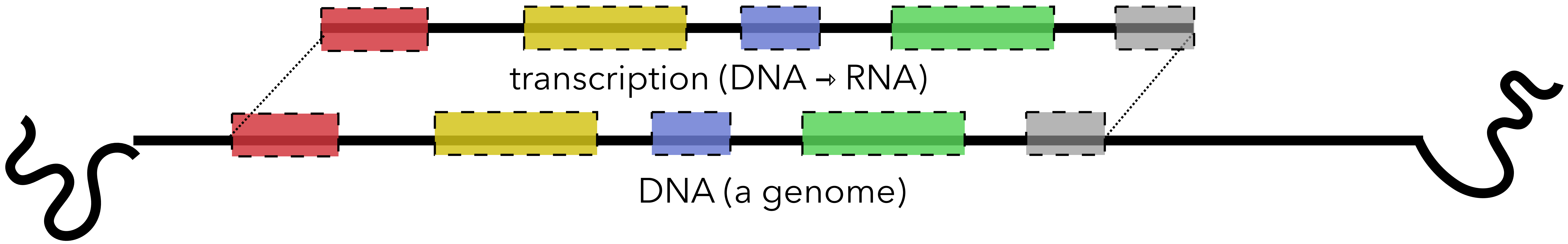
Transcript assembly using RNA-seq data

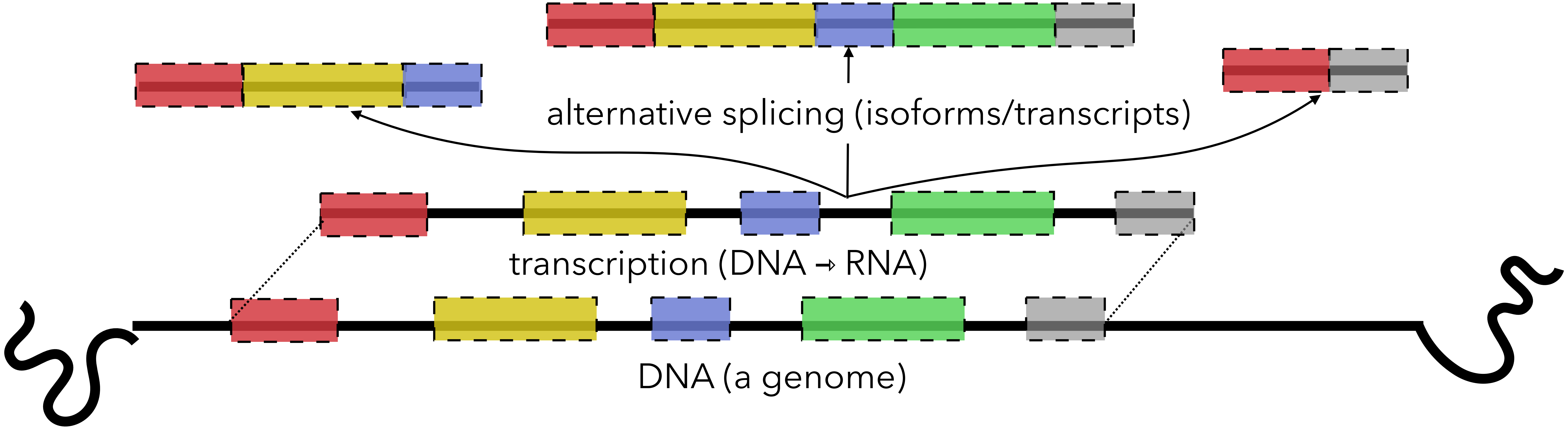
Basics of RNA-seq



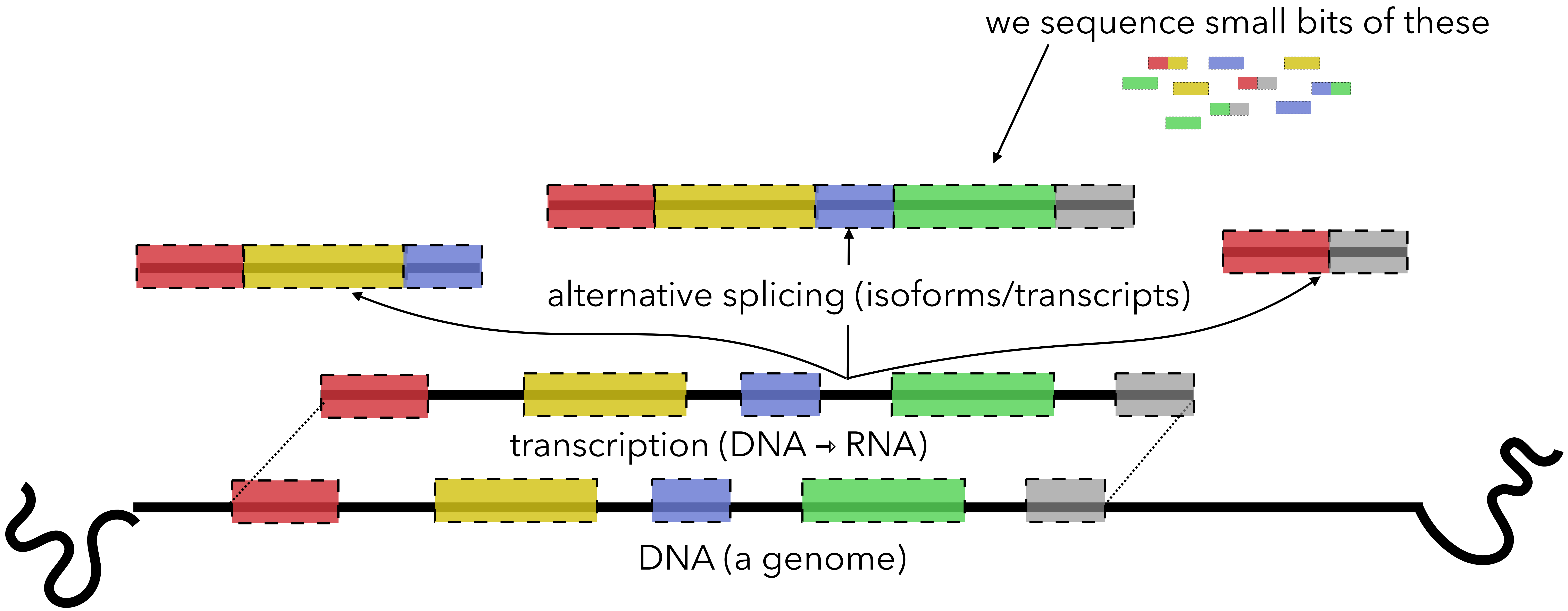
Basics of RNA-seq



Basics of RNA-seq


alternative splicing (isoforms/transcripts)
transcription (DNA → RNA)
DNA (a genome)

Basics of RNA-seq



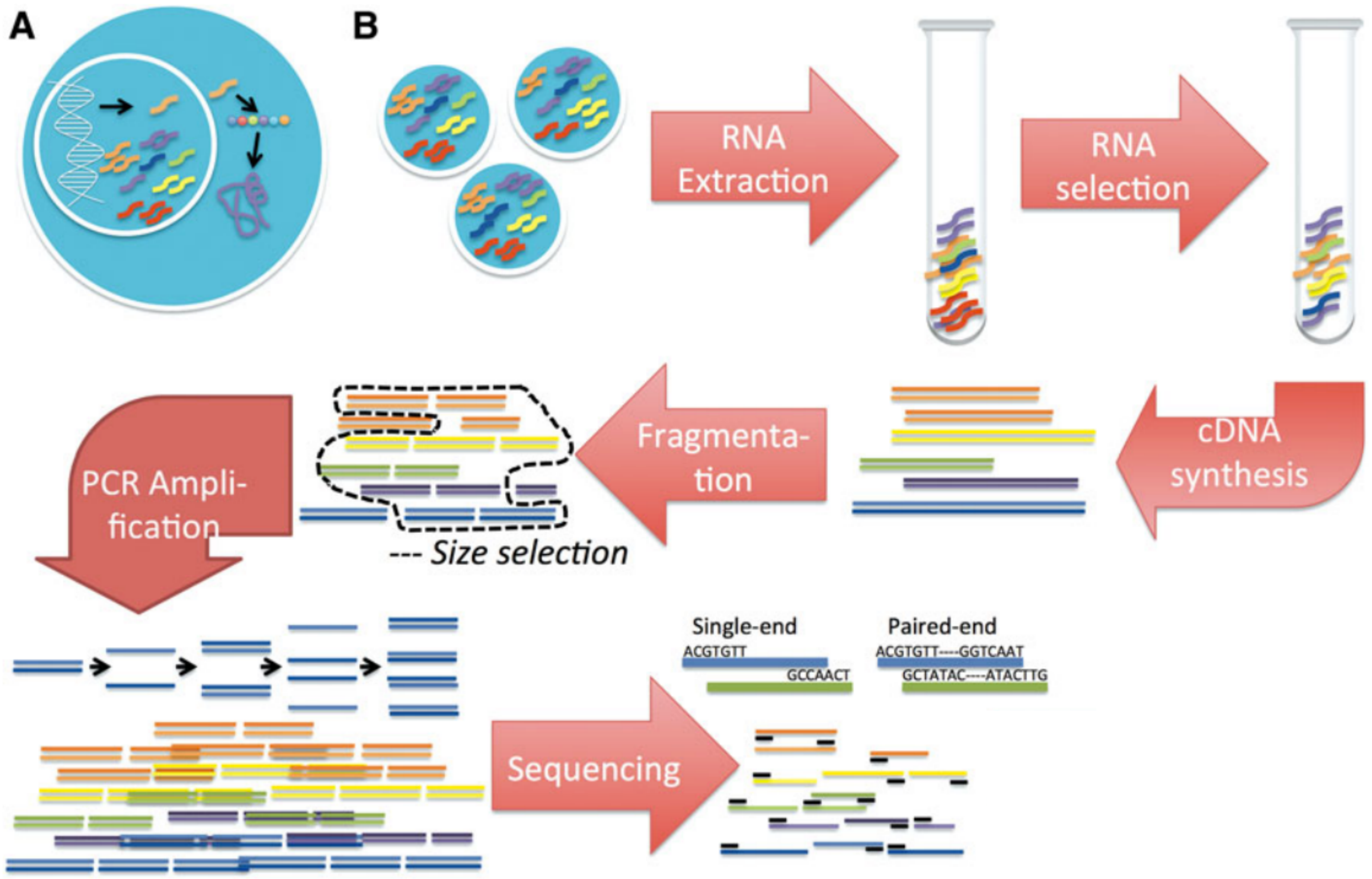
we sequence small bits of these

alternative splicing (isoforms/transcripts)

transcription (DNA → RNA)

DNA (a genome)

Actual protocols are much more involved



Prakash, Celine, and Arndt Von Haeseler. "An Enumerative Combinatorics Model for Fragmentation Patterns in RNA Sequencing Provides Insights into Nonuniformity of the Expected Fragment Starting-Point a
of Computational Biology 24.3 (2017): 200-212.

Many uses of RNA-seq for transcriptome analysis

RNA-seq data has many uses in transcriptome analysis. Some common uses include:

Fusion/chimera detection

Variant (SNP, SV, CNV) detection

Transcript assembly

Genome guided & de novo

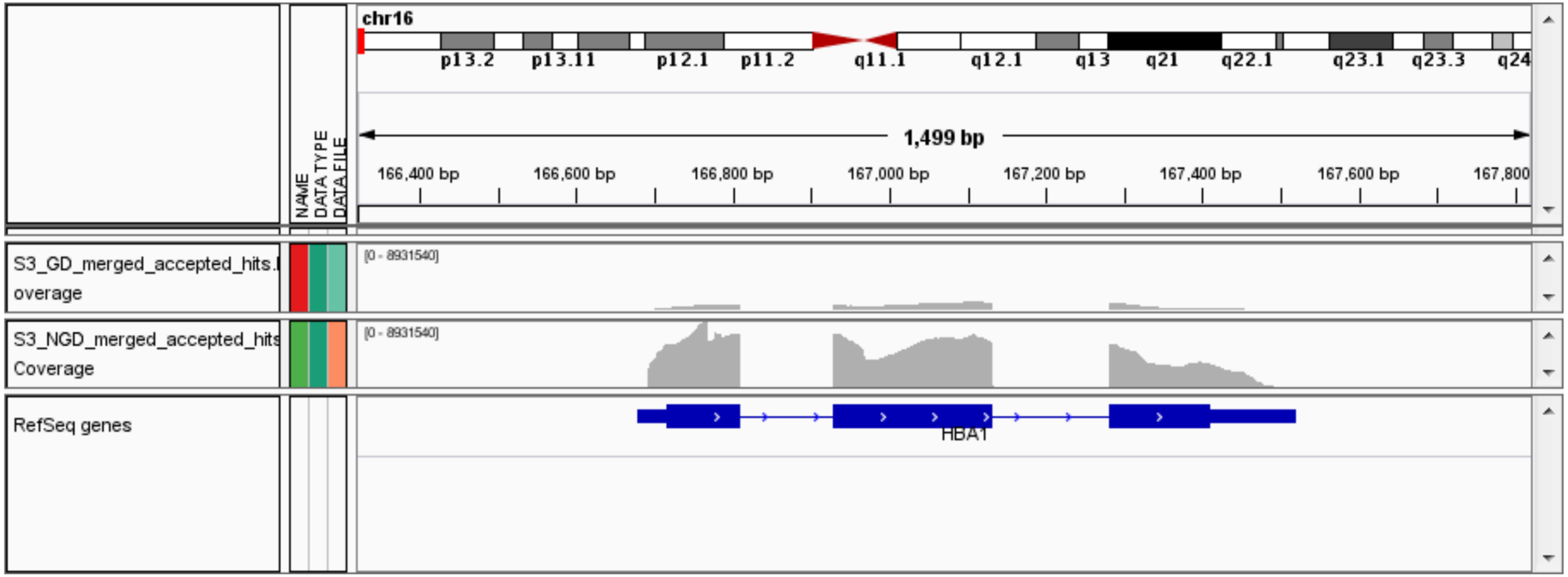
Transcript quantification

Differential expression, alternative splicing analysis

Build higher-level models of transcription

co-expression networks -> regulatory networks

The benefit is reads are drawn directly from transcripts!



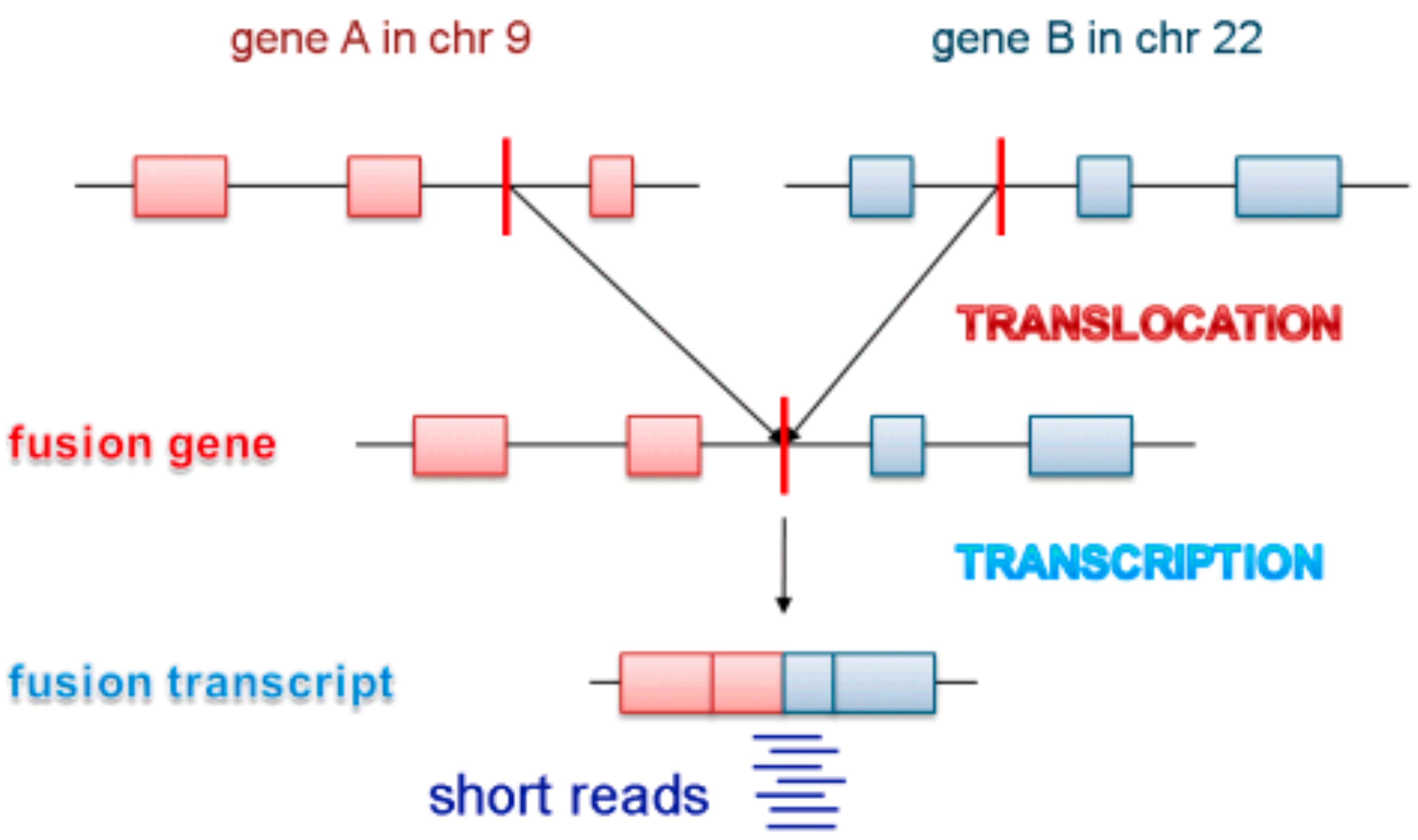
RNA-Seq reads come from a spliced transcript — if we can map them back to the genome, they give us evidence of **transcribed** regions.

Human genome contains > 14,000 pseudogenes [Pei et al. Genome Biology 2012]

Image from: Shin, Heesun, et al. "Variation in RNA-Seq Transcriptome Profiles of Peripheral Whole Blood from Healthy Individuals with and without Globin Depletion." PloS one 9.3 (2014): e91041.

This means you see what is there, not just what is annotated

Fusion/chimera detection



Variant (SNP, SV, CNV) detection

Find small (SNP) or large (SV) variation in how read map back to their genes of origin

Find differences in the number of copies of a gene in the DNA (CNV)

image from: http://biome.ewha.ac.kr:8080/FusionGene/

Many uses of RNA-seq for transcriptome analysis

RNA-seq data has many uses in transcriptome analysis. Some common uses include:

Fusion/chimera detection

Variant (SNP, SV, CNV) detection

Transcript assembly

Genome guided & de novo

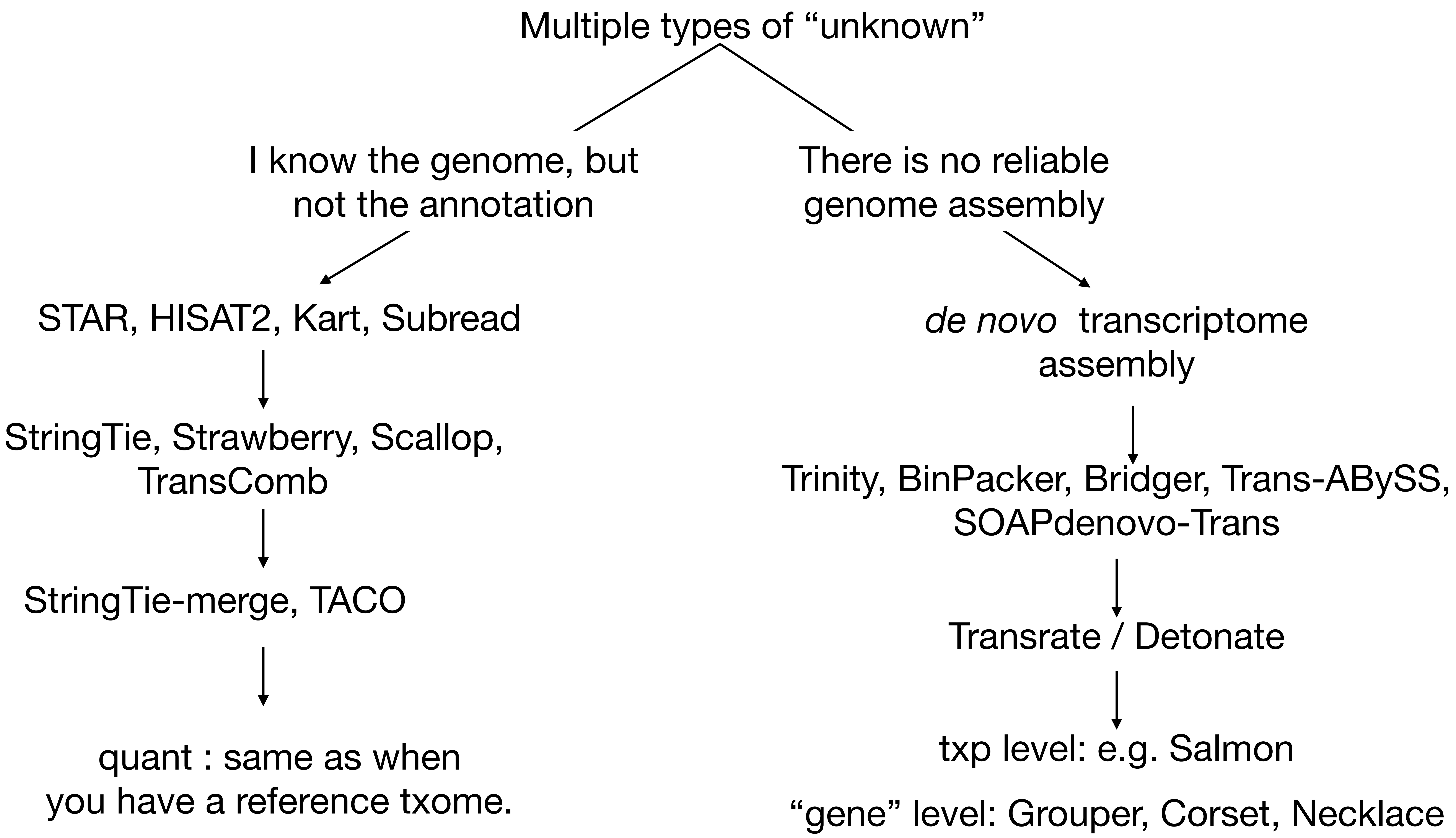
Transcript quantification

Differential expression, alternative splicing analysis

Build higher-level models of transcription

co-expression networks -> regulatory networks

Into the Unknown : What to do when you don’t have / trust your reference



Multiple types of “unknown”

I know the genome, but not the annotation

STAR, HISAT2, Kart, Subread

StringTie, Strawberry, Scallop, TransComb

StringTie-merge, TACO

quant : same as when you have a reference txome.

There is no reliable genome assembly

de novo transcriptome assembly

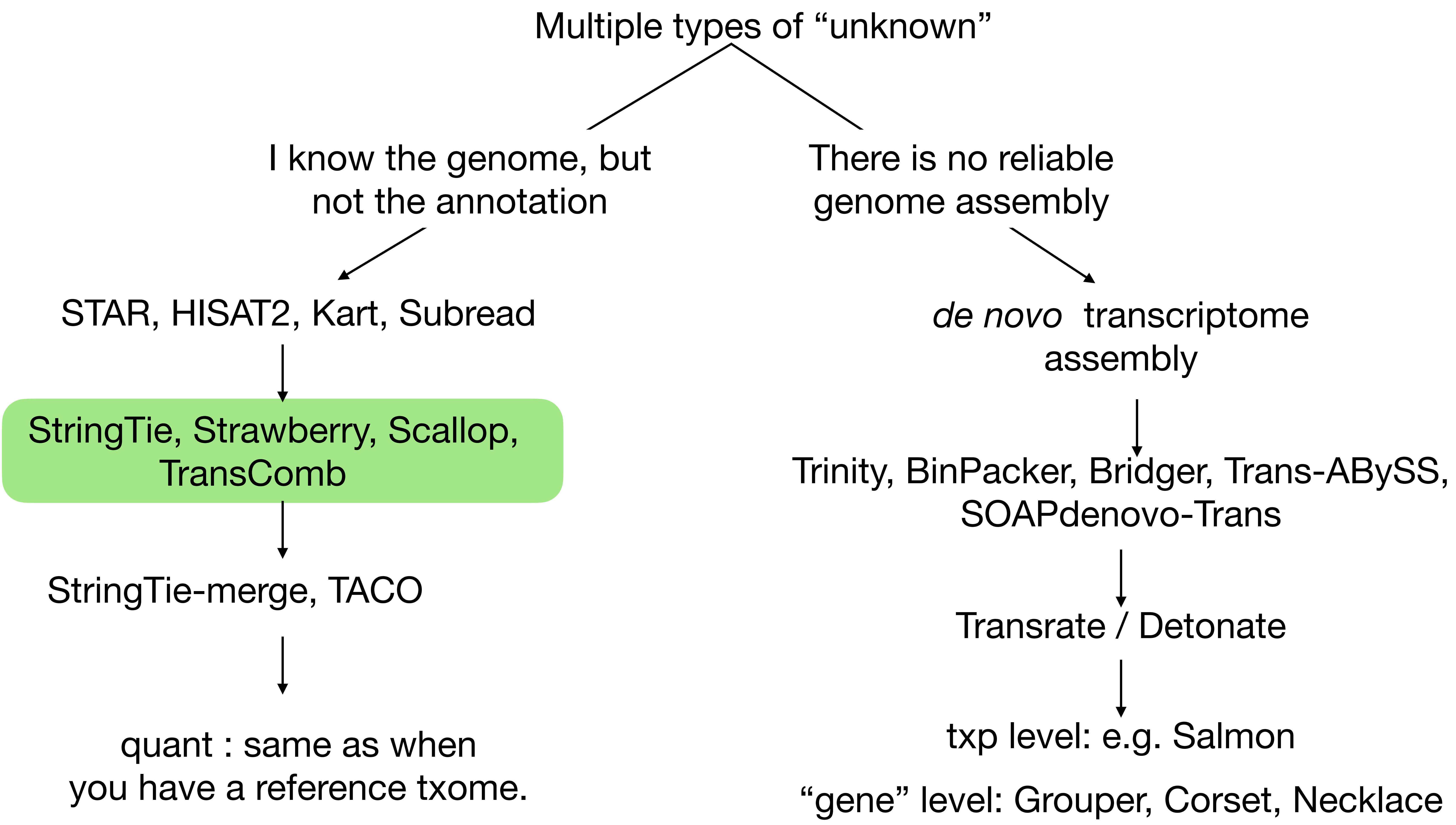
Trinity, BinPacker, Bridger, Trans-ABySS, SOAPdenovo-Trans

Transrate / Detonate

txp level: e.g. Salmon

“gene” level: Grouper, Corset, Necklace

Into the Unknown : What to do when you don’t have / trust your reference



Multiple types of “unknown”

I know the genome, but not the annotation

STAR, HISAT2, Kart, Subread

StringTie, Strawberry, Scallop, TransComb

StringTie-merge, TACO

quant : same as when you have a reference txome.

There is no reliable genome assembly

de novo transcriptome assembly

Trinity, BinPacker, Bridger, Trans-ABySS, SOAPdenovo-Trans

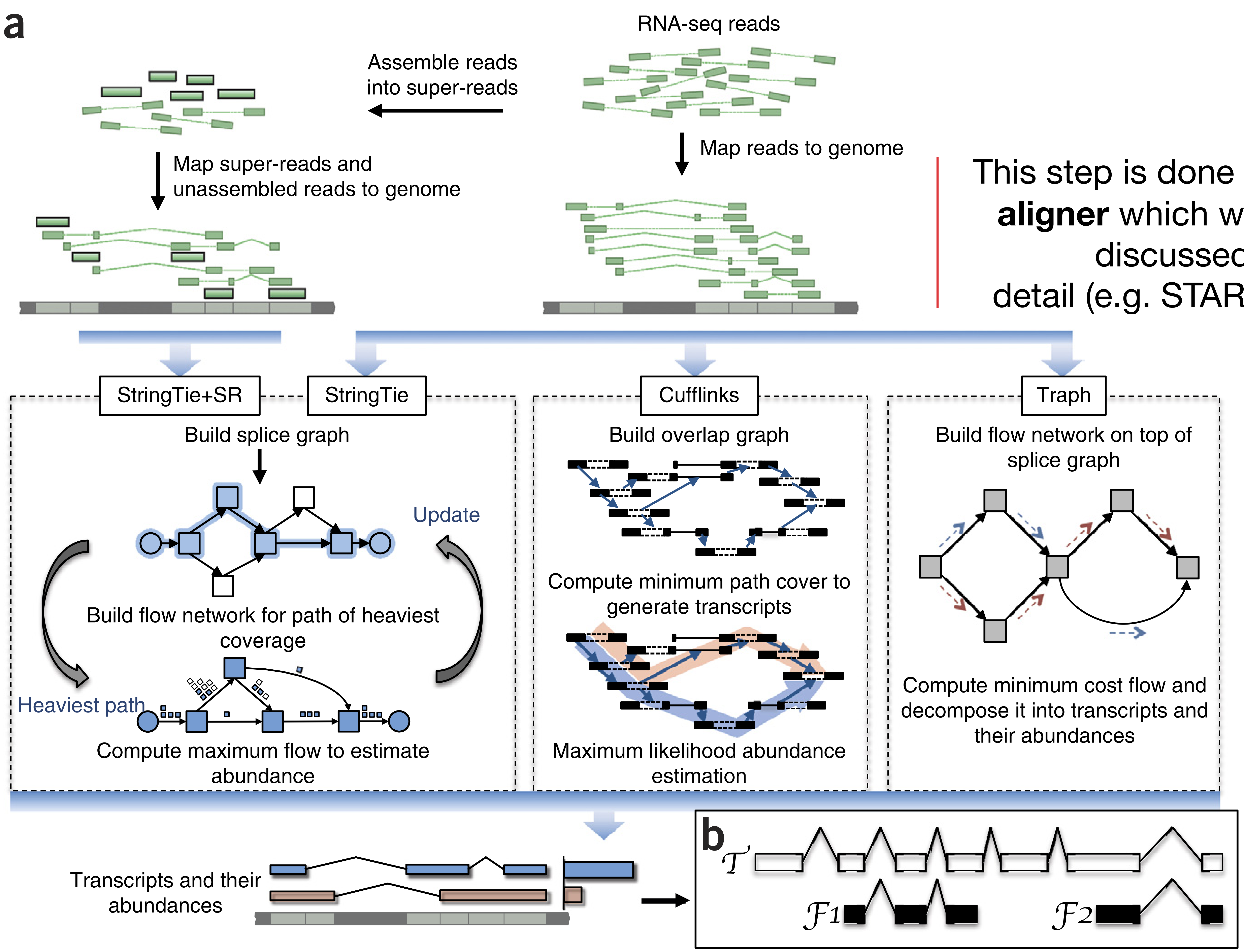
Transrate / Detonate

txp level: e.g. Salmon

“gene” level: Grouper, Corset, Necklace

I’ll focus mostly on *reference-guided* assembly.

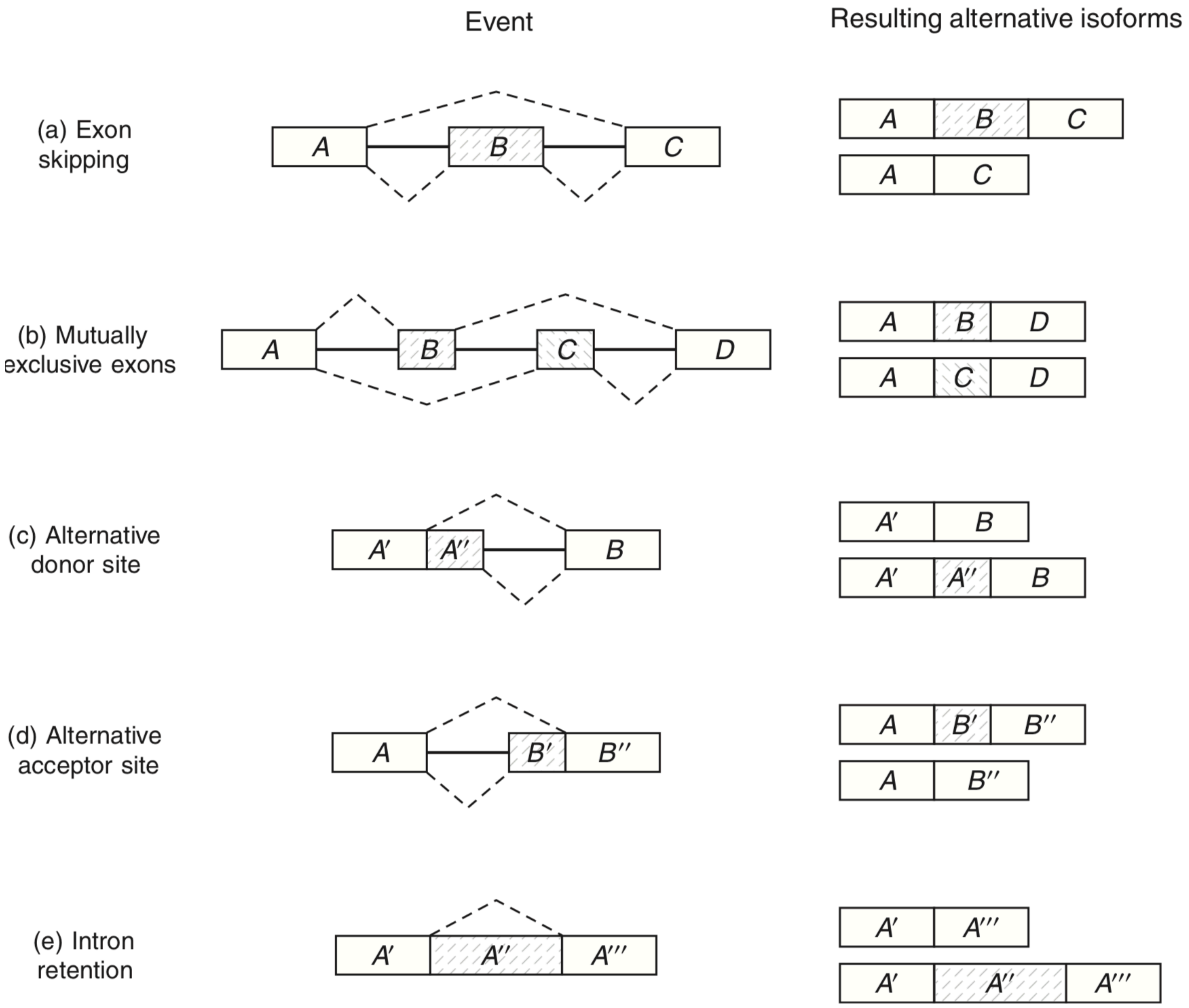
Outline of transcript assembly workflow


a
RNA-seq reads
Assemble reads into super-reads
Map reads to genome
Map super-reads and unassembled reads to genome
StringTie+SR
StringTie
Cufflinks
Traph
Build splice graph
Update
Build flow network for path of heaviest coverage
Heaviest path
Compute maximum flow to estimate abundance
Build overlap graph
Compute minimum path cover to generate transcripts
Maximum likelihood abundance estimation
Build flow network on top of splice graph
Compute minimum cost flow and decompose it into transcripts and their abundances
Transcripts and their abundances
b
$\mathcal{T}$
$\mathcal{F}1$
$\mathcal{F}2$

This step is done by **spliced aligner** which we haven't discussed in detail (e.g. STAR/ HISAT2)

Pertea, Mihaela, et al. "StringTie enables improved reconstruction of a transcriptome from RNA-seq reads." *Nature biotechnology* 33.3 (2015): 290.

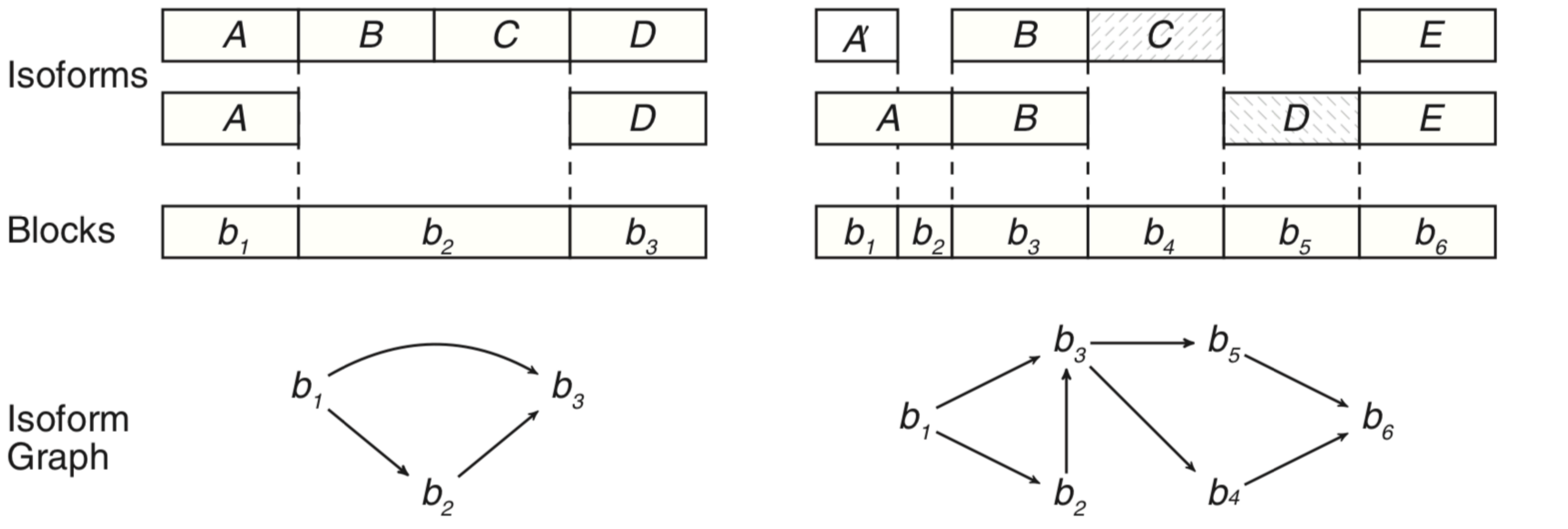
A detour: The splicing graph



Beretta, Stefano, et al. "Modeling alternative splicing variants from RNA-Seq data with isoform graphs." *Journal of Computational Biology* 21.1 (2014): 16-40.

A detour: The splicing graph

In reality we observe coverage by reads, not exons. Therefore, we end up building a slightly different (data-dependent) graph.

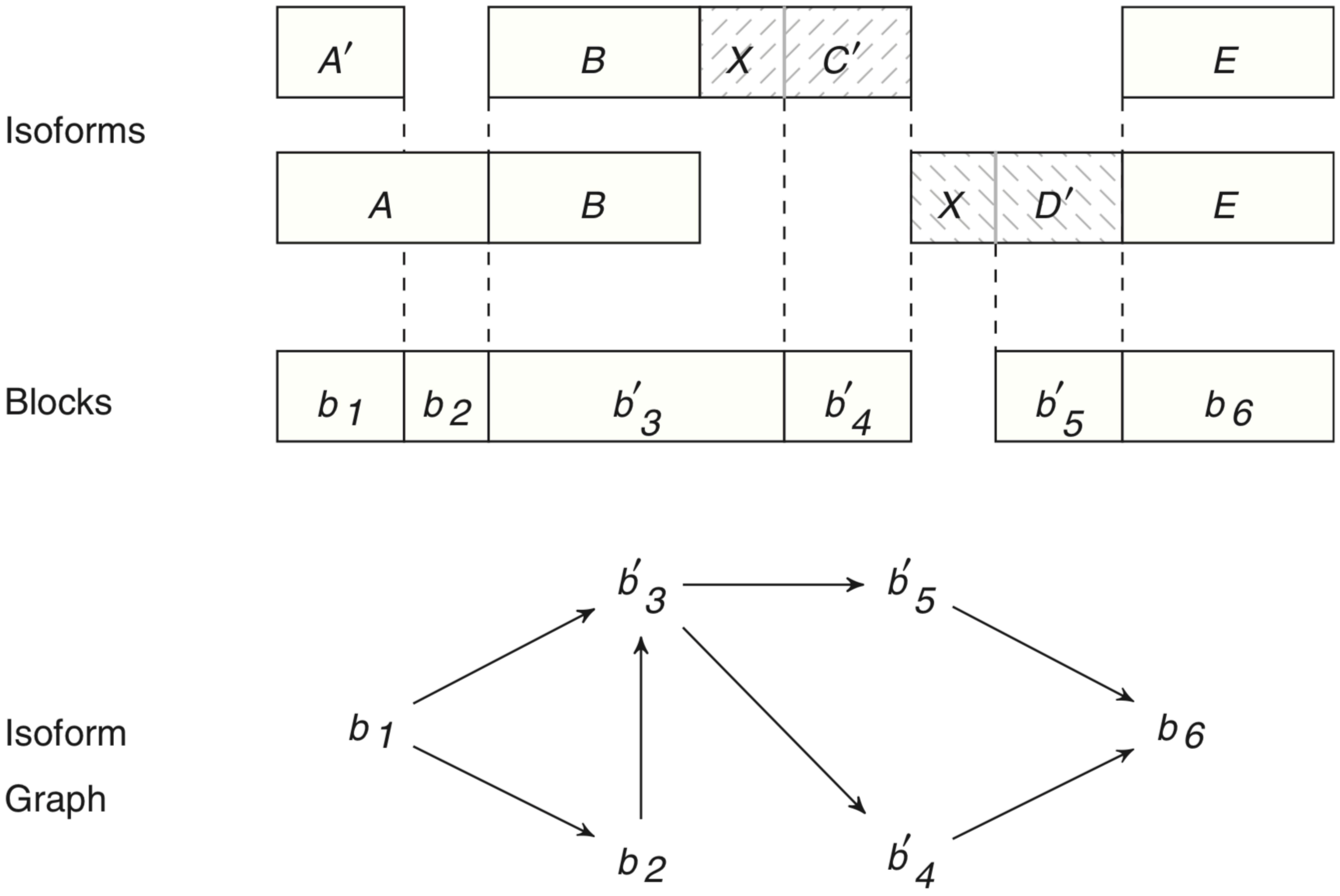


(a) Exon skipping.

(b) Alternative donor sites and mutually exclusive exons.

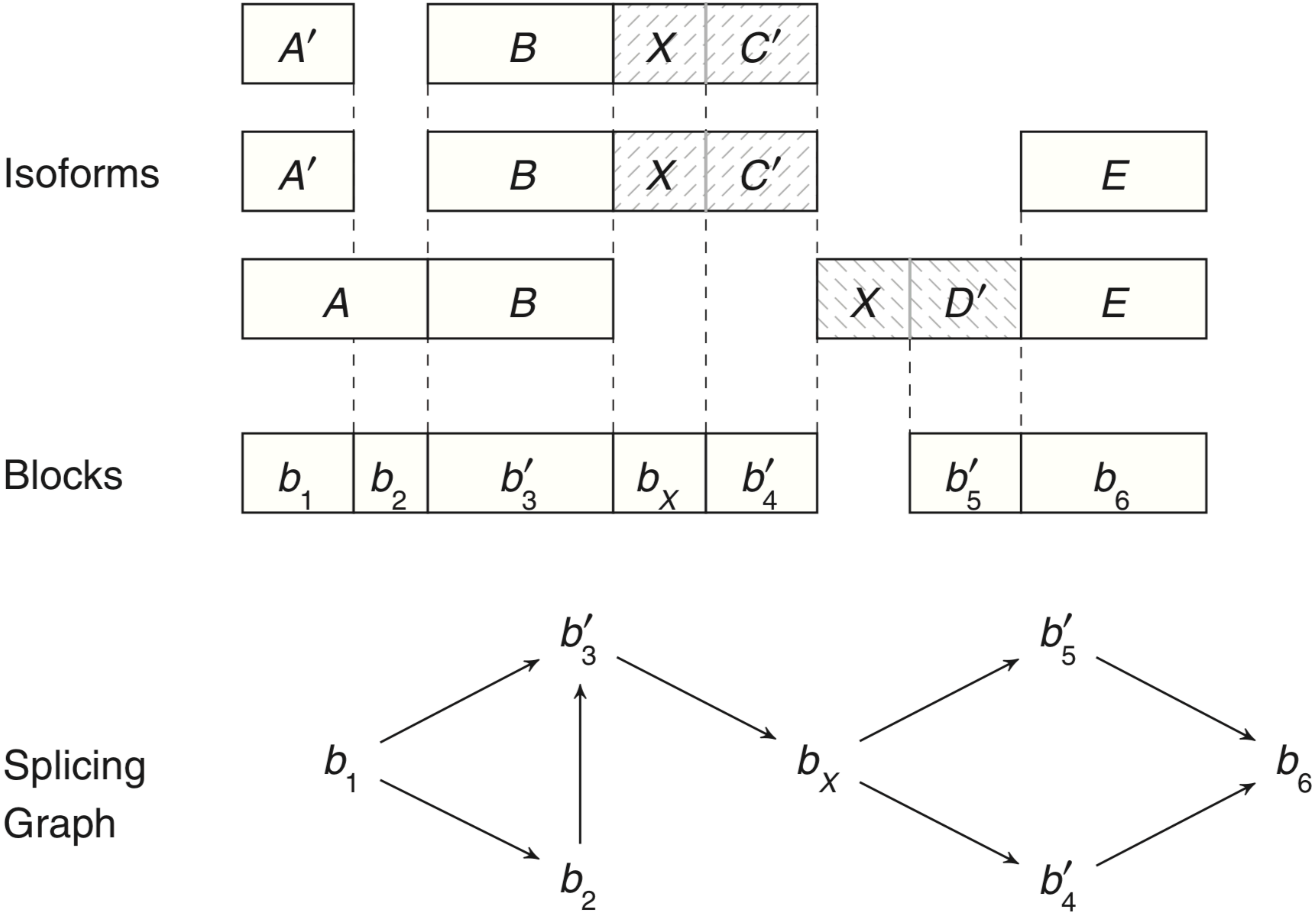
Beretta, Stefano, et al. "Modeling alternative splicing variants from RNA-Seq data with isoform graphs." *Journal of Computational Biology* 21.1 (2014): 16-40.

A detour: The splicing graph



Beretta, Stefano, et al. "Modeling alternative splicing variants from RNA-Seq data with isoform graphs." *Journal of Computational Biology* 21.1 (2014): 16-40.

A detour: The splicing graph



Beretta, Stefano, et al. "Modeling alternative splicing variants from RNA-Seq data with isoform graphs." *Journal of Computational Biology* 21.1 (2014): 16-40.

Building of Splice Graph

StringTie builds an alternative splicing graph (ASG) from all reads at a genomic locus.

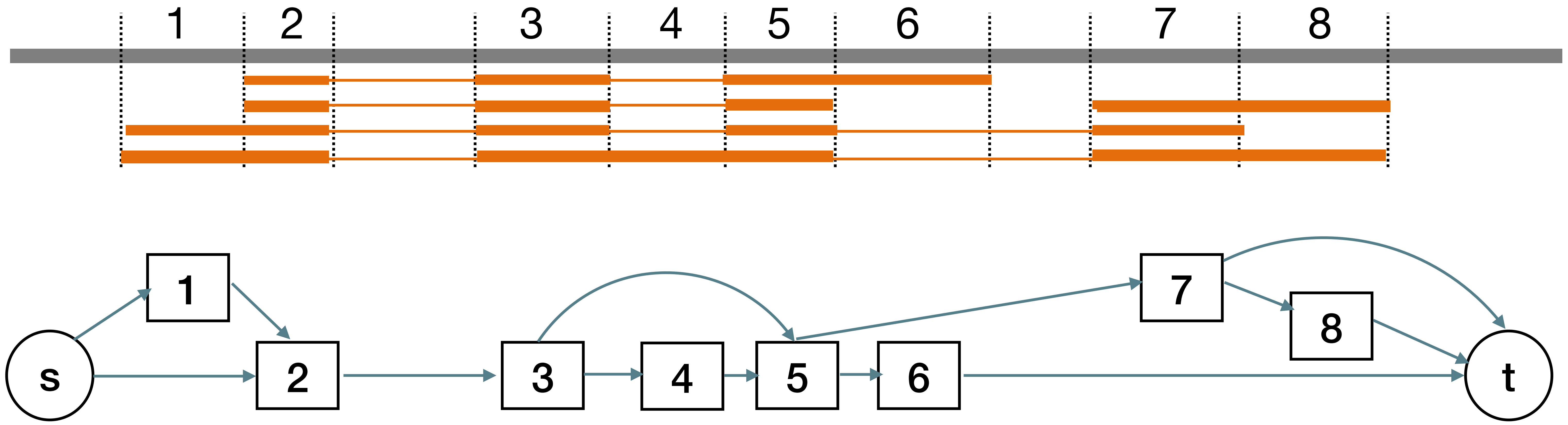
Skips this locus if too many (>95%) of the reads here are multi-mapping

Otherwise, reads are naively given 1/k mass at each of their k multi-mapping loci.

Splice graph is a DAG where nodes are contiguous genomic regions not interrupted by spliced alignments, and edges are placed between two nodes between which a spliced alignment occurs.

Building of Splice Graph

ASG example (adapted from supp fig. 1)



Pertea, Mihaela, et al. "StringTie enables improved reconstruction of a transcriptome from RNA-seq reads." *Nature biotechnology* 33.3 (2015): 290.

Processing the Splice Graph

StringTie identifies transcripts using the ASG with the following iterative process:

1. Heuristically choose a “heavy” path (a path with the heaviest node) in the ASG

2. Estimate path expression by computing max-flow in a flow graph corresponding to this sub-path of the ASG. Subtract the read mass assigned to the nodes in this path & repeat.

Choosing a Heaviest Path

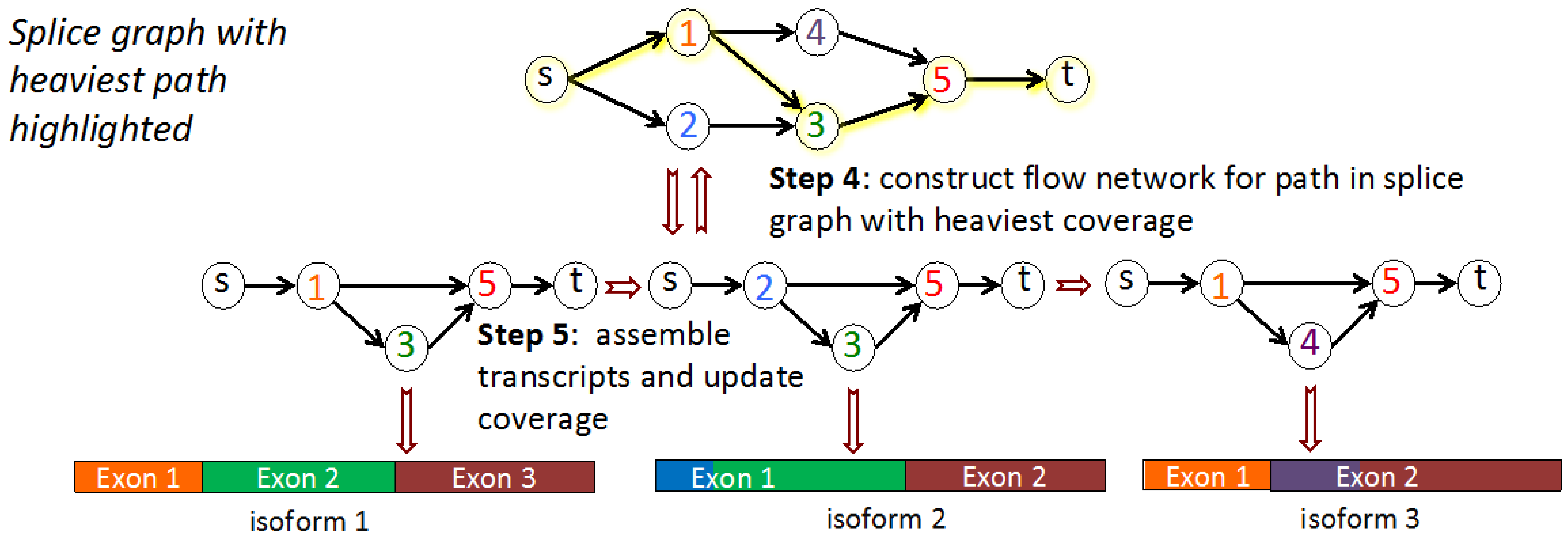
StringTie chooses the heaviest path greedily, as follows (this is an O(n) algorithm):

Pick the ASG node with the highest per-base read coverage.

Extend nodes to the **source** by adding to the path the adjacent node with the largest # of compatible read fragments.

Extend nodes to the **sink** by adding to the path the adjacent node with the largest # of compatible read fragments.

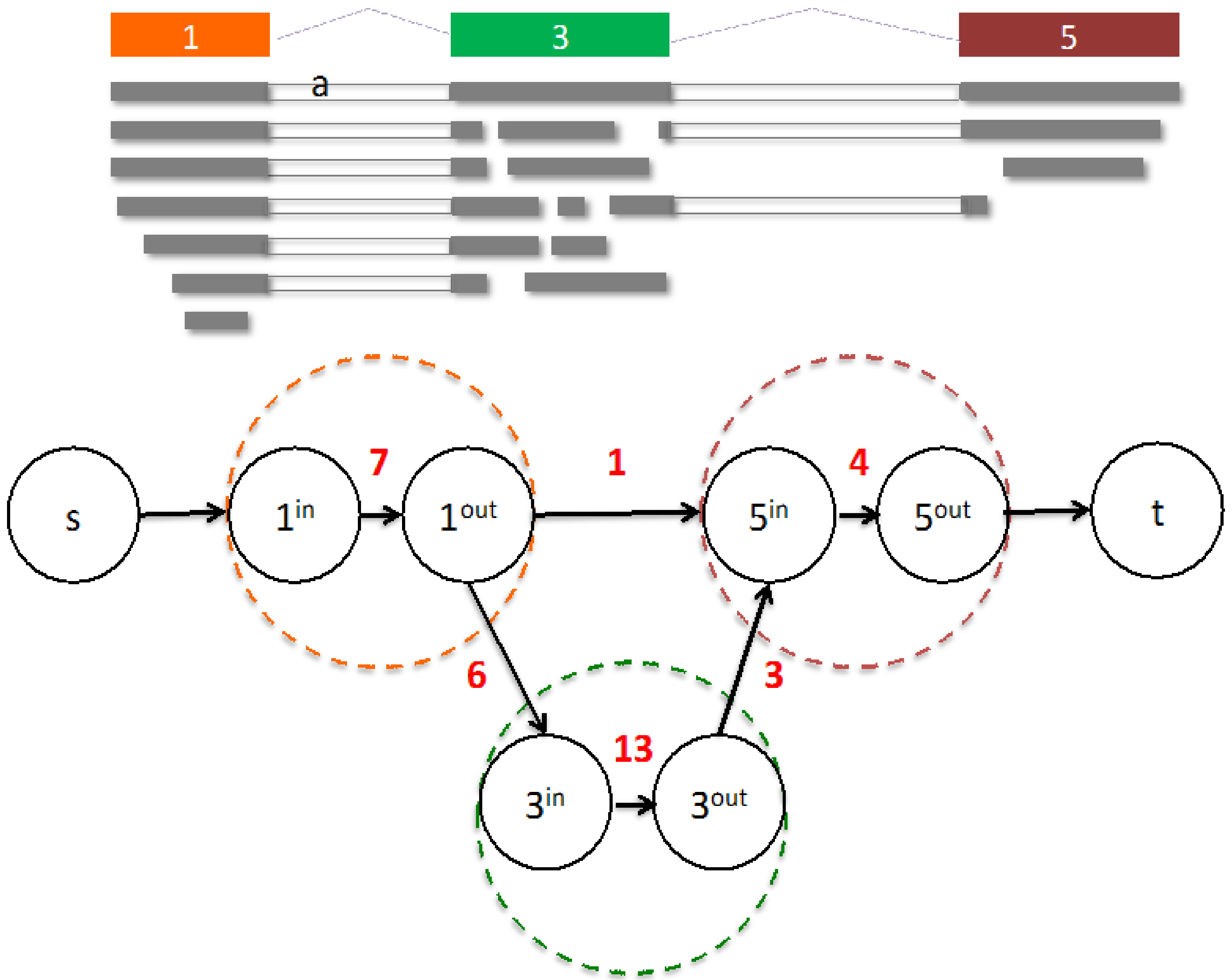
Constructing the Flow Network


Splice graph with heaviest path highlighted
s
1
2
3
4
5
t
Step 4: construct flow network for path in splice graph with heaviest coverage
Step 5: assemble transcripts and update coverage
Exon 1
Exon 2
Exon 3
isoform 1
Exon 1
Exon 2
isoform 2
Exon 1
Exon 2
isoform 3

Note: The flow network is constructed separately for each selected transcript — the network on which the flow problem is solved does *not* correspond to the entire ASG!

Pertea, Mihaela, et al. "StringTie enables improved reconstruction of a transcriptome from RNA-seq reads." *Nature biotechnology* 33.3 (2015): 290.

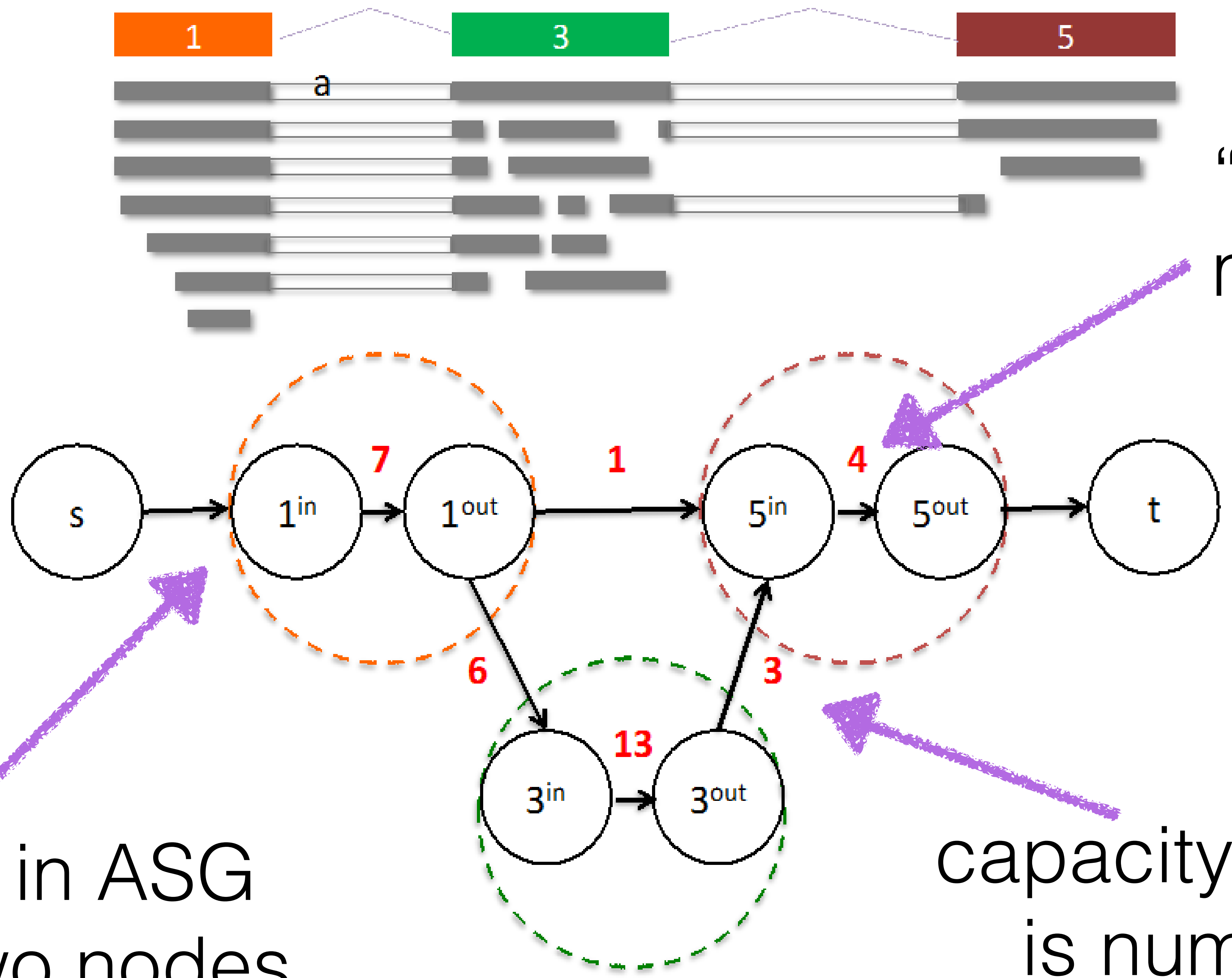
Constructing the Flow Network



Supplementary Figure 13. Flow network associated with a transcript (shown with colored nodes). 15 fragments (shown in grey) align to the transcript. Two nodes in the flow network are connected iff a fragment starts and ends at those nodes. E.g., nodes 1 and 5 are connected because fragment (a) starts at node 1 and ends at node 5. For each colored node in the transcript, two nodes are created in the flow network. Capacities on edges (not connecting source or sink) are shown in red.

Pertea, Mihaela, et al. "StringTie enables improved reconstruction of a transcriptome from RNA-seq reads." *Nature biotechnology* 33.3 (2015): 290.

Constructing the Flow Network



“within” edges may also have a bias factor

$$b_v = \frac{\sum_{u \in ASG} w(f_{uv})}{\sum_{u \in ASG} w(f_{vu})}$$

each node in ASG becomes two nodes in / out in flow network

capacity of “between” edges is number of alignments starting in one node and ending in the other

Supplementary Figure 13. Flow network associated with a transcript (shown with colored nodes). 15 fragments (shown in grey) align to the transcript. Two nodes in the flow network are connected iff a fragment starts and ends at those nodes. E.g., nodes 1 and 5 are connected because fragment (a) starts at node 1 and ends at node 5. For each colored node in the transcript, two nodes are created in the flow network. Capacities on edges (not connecting source or sink) are shown in red.

Pertea, Mihaela, et al. "StringTie enables improved reconstruction of a transcriptome from RNA-seq reads." *Nature biotechnology* 33.3 (2015): 290.

Constructing the Flow Network

Given the flow network, StringTie solves a *generalized* max-flow problem

generalized — edges may have multipliers (bias factors), so flow may be gained or lost as it is sent through the network.

$N = (G, s, t, c)$, where $G = (V, E)$, with $E \subseteq V \times V$. ← Flow Network

$s \in V$ and $t \in V$ ← Source / Sink

$c : E \rightarrow R^+$ ← Edge capacities

$f : E \rightarrow R^+$ ←

$||f|| = \Sigma_{(v,\, t) \in E} f((v, t))$ ←

Find a **flow** that is **maximum**

$$(1)\quad 0 \leq f((u, v)) \leq c\,((u, v)), \text{ for every } (u, v) \in E,$$

$$(2)\quad \Sigma_{(u,\, v) \in E} f((u, v)) = \Sigma_{(v,\, u) \in E} f((v, u)), \text{ for every } v \in V, v \neq s, t.$$

$$l_{(u,v)} \leq f((u,v)) \leq c((u,v)), \quad \text{for every } (u,v) \in E$$

Satisfying

Constructing the Flow Network

Given the flow network, StringTie solves a *generalized* max-flow problem

generalized — edges may have multipliers (bias factors), so flow may be gained or lost as it is sent through the network.

$N = (G, s, t, c)$, where $G = (V, E)$, with $E \subseteq V \times V$. ← Flow Network

$s \in V$ and $t \in V$ ← Source / Sink

$c : E \rightarrow R^+$ ← Edge capacities

$f : E \rightarrow R^+$ ← Find a **flow** that is **maximum**

$||f|| = \Sigma_{(v, t) \in E} f((v, t))$

(1) $0 \leq f((u, v)) \leq c((u, v))$, for every $(u, v) \in E$,

(2) $\Sigma_{(u, v) \in E} f((u, v)) = \Sigma_{(v, u) \in E} f((v, u))$, for every $v \in V$, $v \neq s, t$.

$l_{(u,v)} \leq f((u,v)) \leq c((u,v))$, for every $(u,v) \in E$

Satisfying

StringTie solves this generalized max-flow problem using an augmenting path algorithm

Recall: Max Flow

Flow network: G = (V, E, s, t, c)

Find a flow f : E →R+ of maximum value

Subject to:

(1) Capacity : $0 \leq f(e) \leq c_e$ for all $e \in E$

(2) Conservation : For every $v \in V$ (apart from s and t)

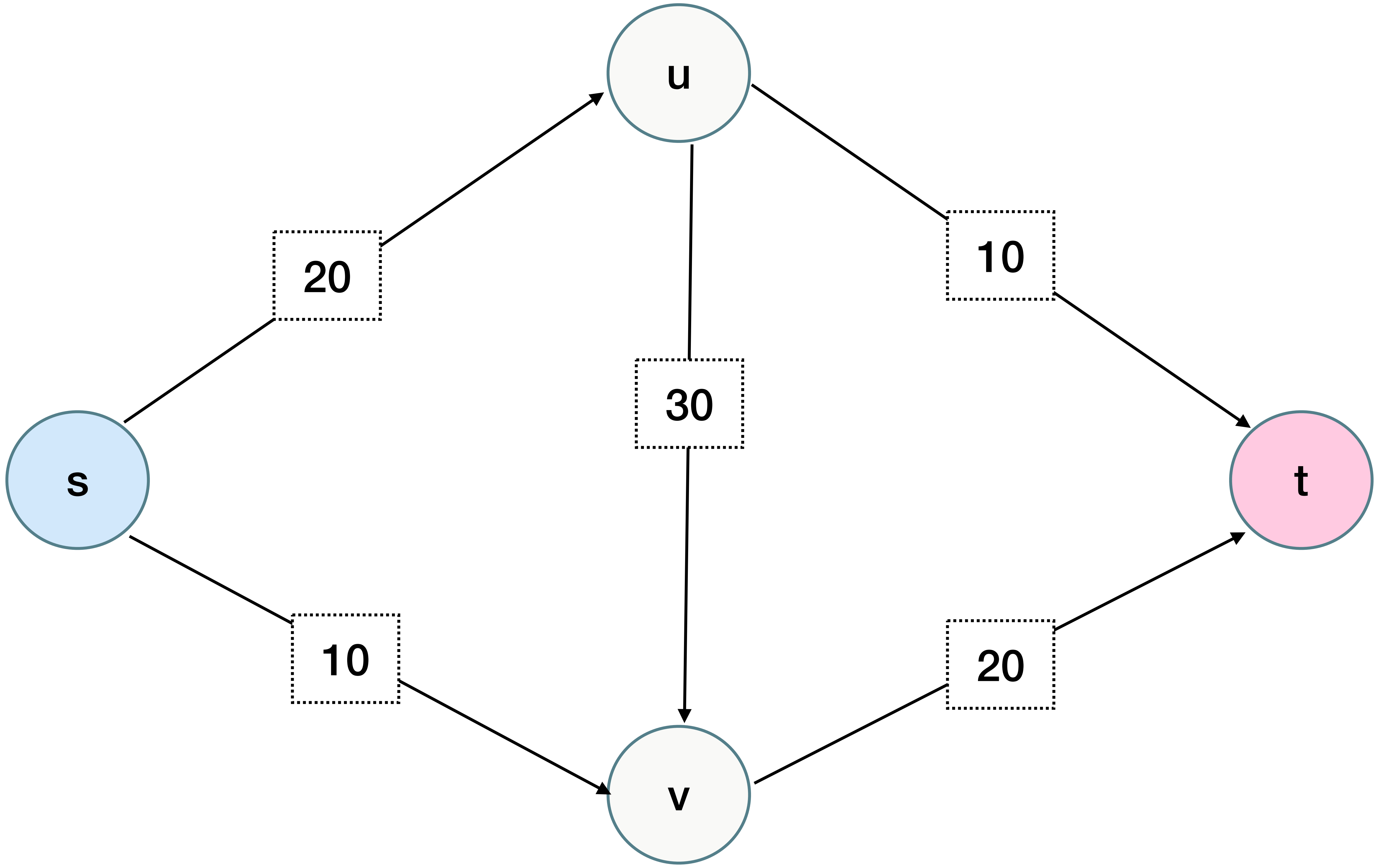
$$\sum_{e \text{ into } v} f(e) = \sum_{e' \text{ out of } v} f(e')$$

Value of the flow is given by:

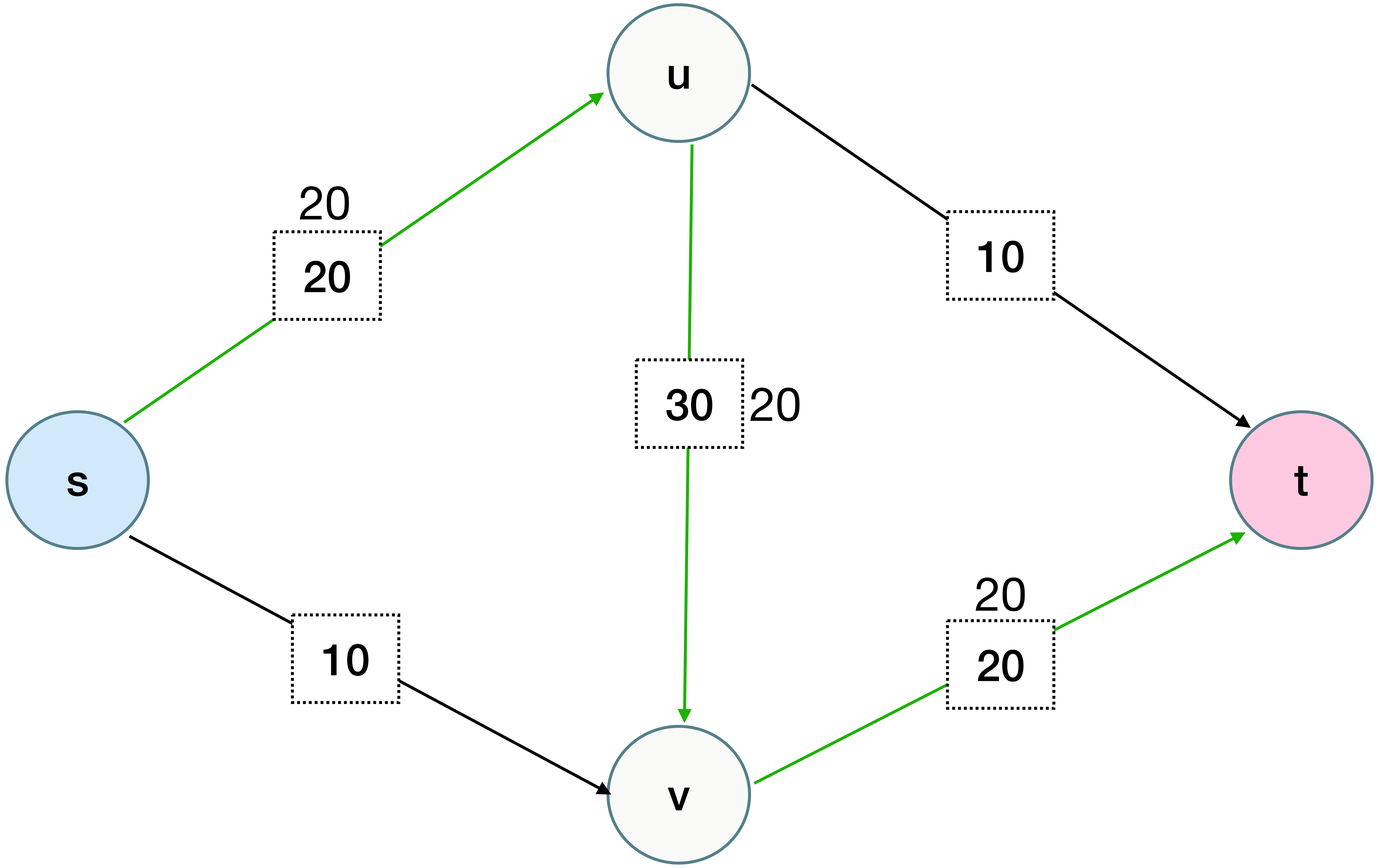
$$v(f) = \sum_{e \textbf{ out of } s} f(e)$$

The below slides follow 7.1 in Kleinberg & Tardos

Recall: Basic Algorithm

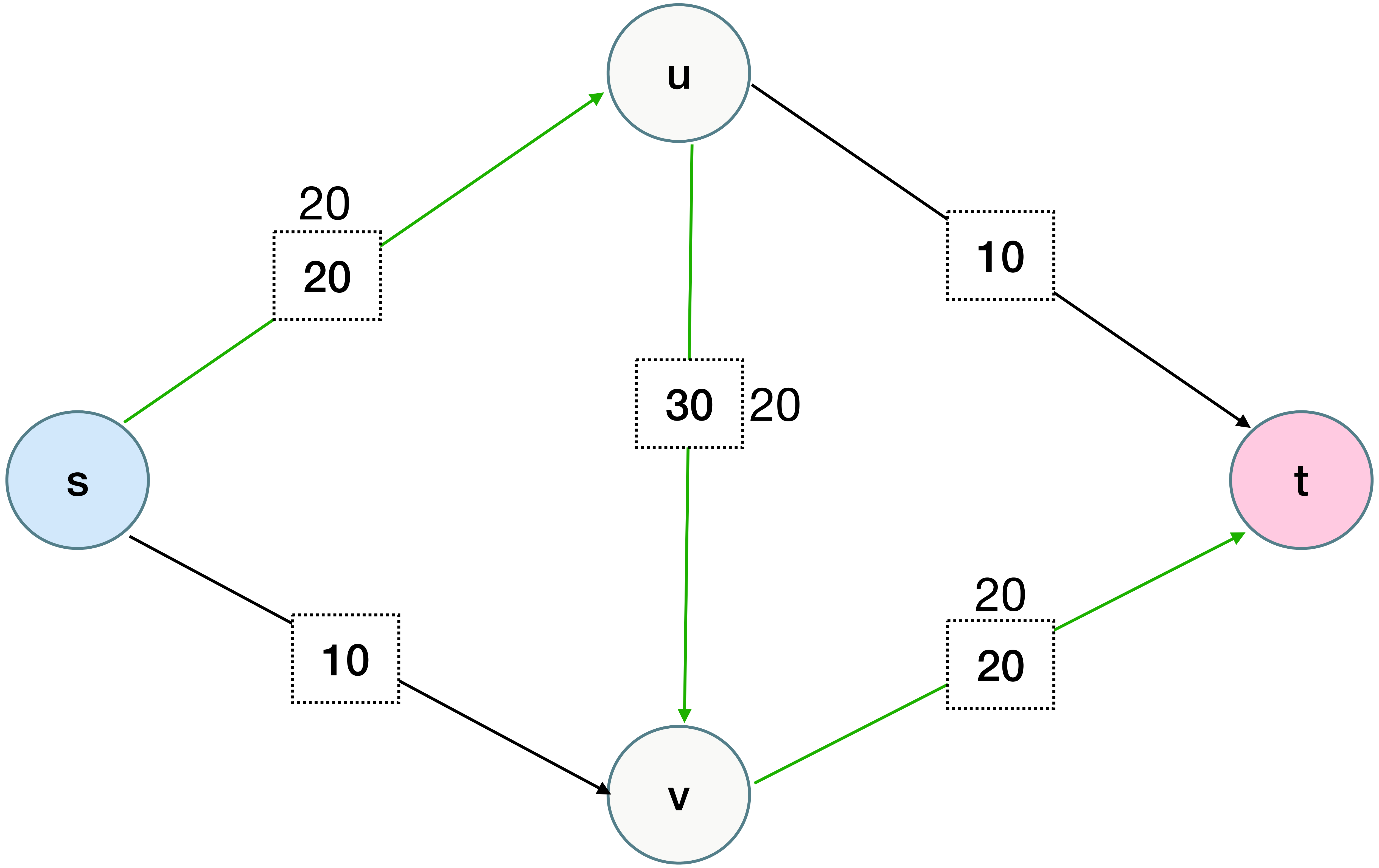


Recall: Basic Algorithm



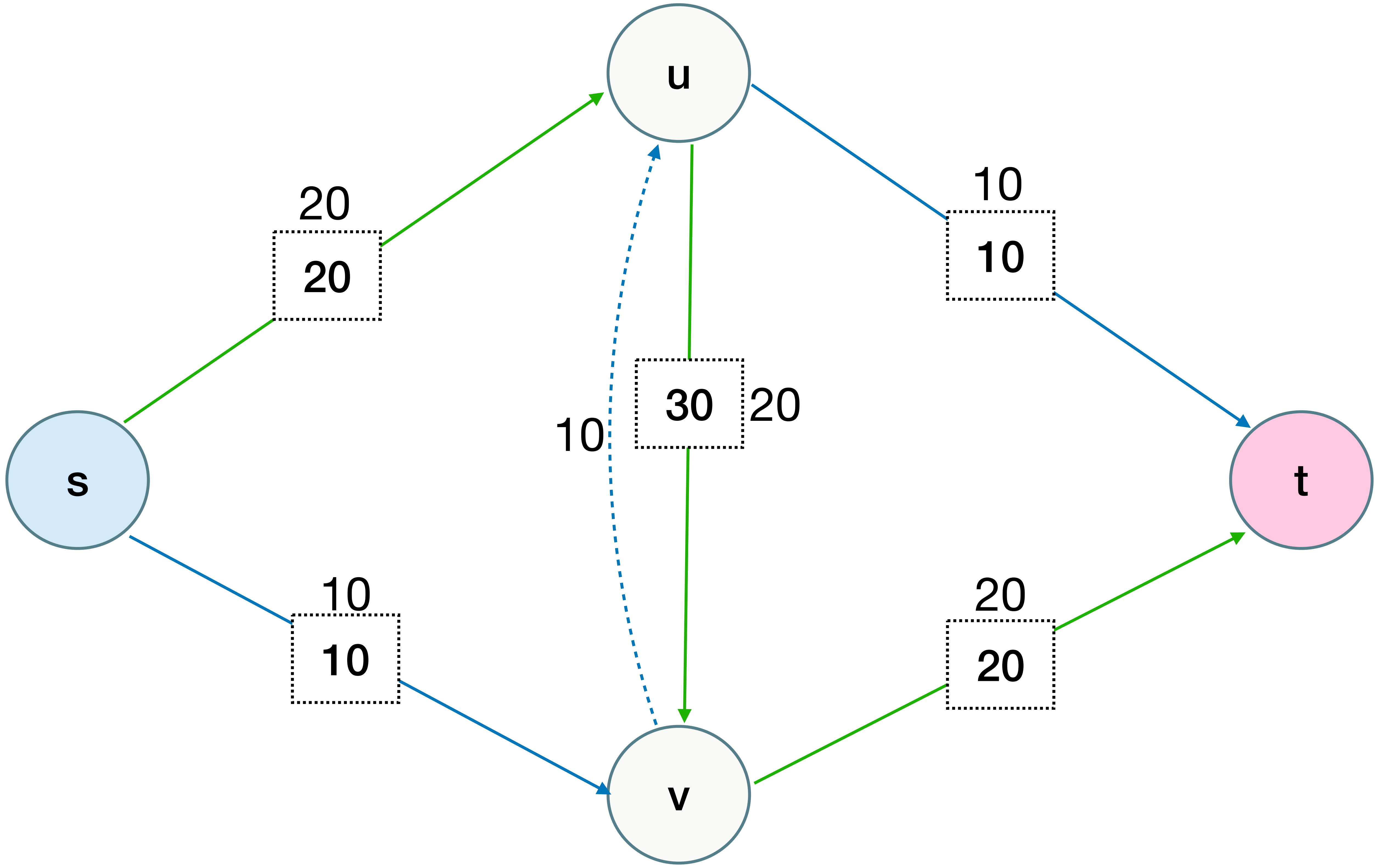
This achieves a flow of value 20, which respects (1) and (2). Is it maximum?

Recall: Basic Algorithm



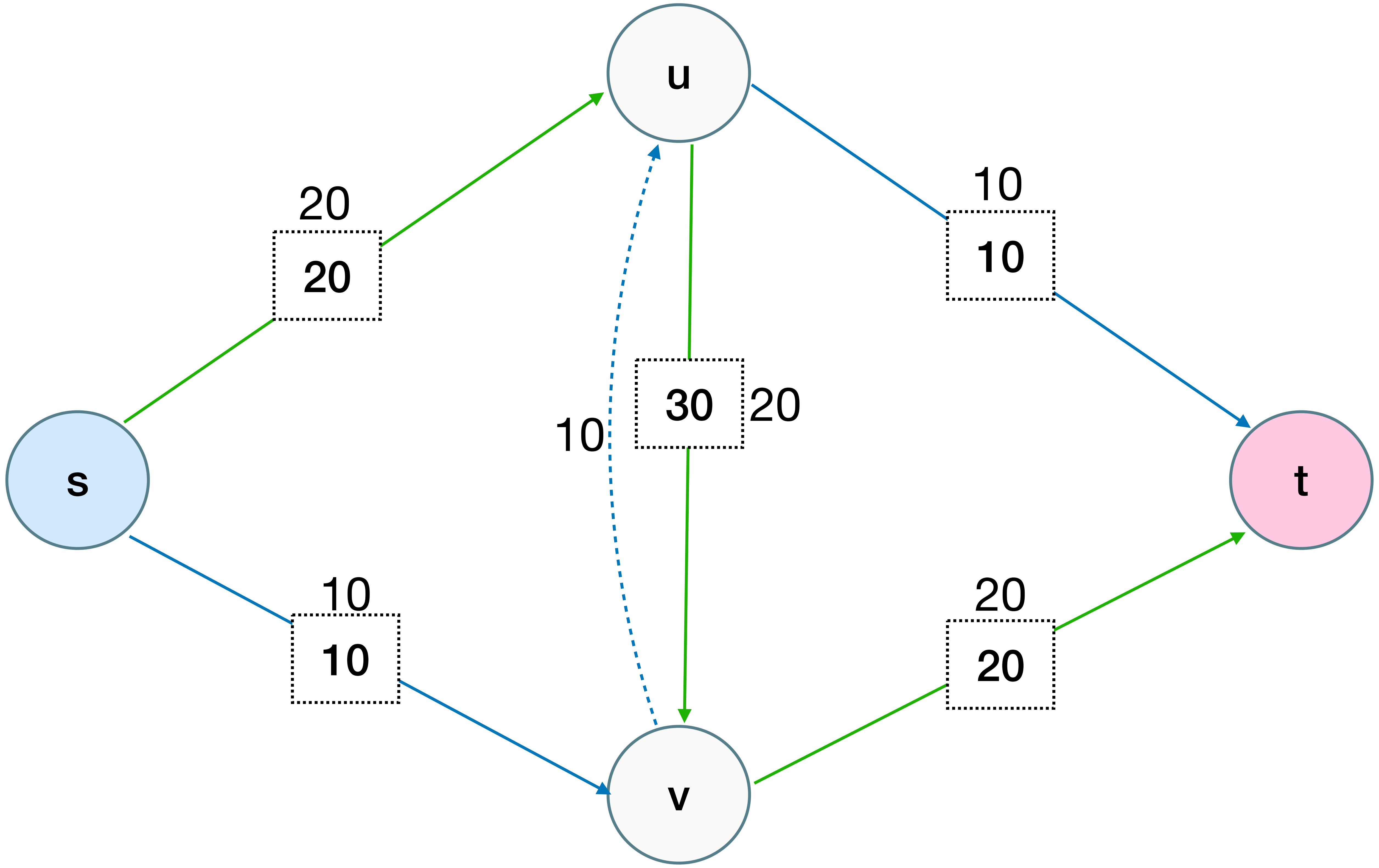
This achieves a flow of value 20, which respects (1) and (2). Is it maximum?
No, but now we are “stuck” by the edges we chose. What if we could “undo” some of the flow?

Recall: Basic Algorithm



This achieves a flow of value 20, which respects (1) and (2). Is it maximum? No, but now we are “stuck” by the edges we chose. What if we could “undo” some of the flow?

Recall: Basic Algorithm



The “dotted” line here is a “backward” edge — it doesn’t exist in the original graph. But flows realized using such residual edges can always be realized in the original graph by changing the forward flows. This leads to the formal idea of the residual graph.

Recall: Basic Algorithm

If G is a flow network with a valid flow f, then the residual

Graph G_f:

Has the same node set as G

for each e = (u,v) in G where $f(e) < c_e$, G_f has an edge e=(u,v) with capacity given by $c_e - f(e)$.

for each e = (u,v) in G where $f(e) > 0$, G_f has an edge e'=(v,u) with capacity f(e) — these are "backward edges"

Recall: Basic Algorithm

Let **bottleneck**(P,f) for a simple s-t path P with flow f be the minimum residual capacity of any edge on P. We define the following subroutine:

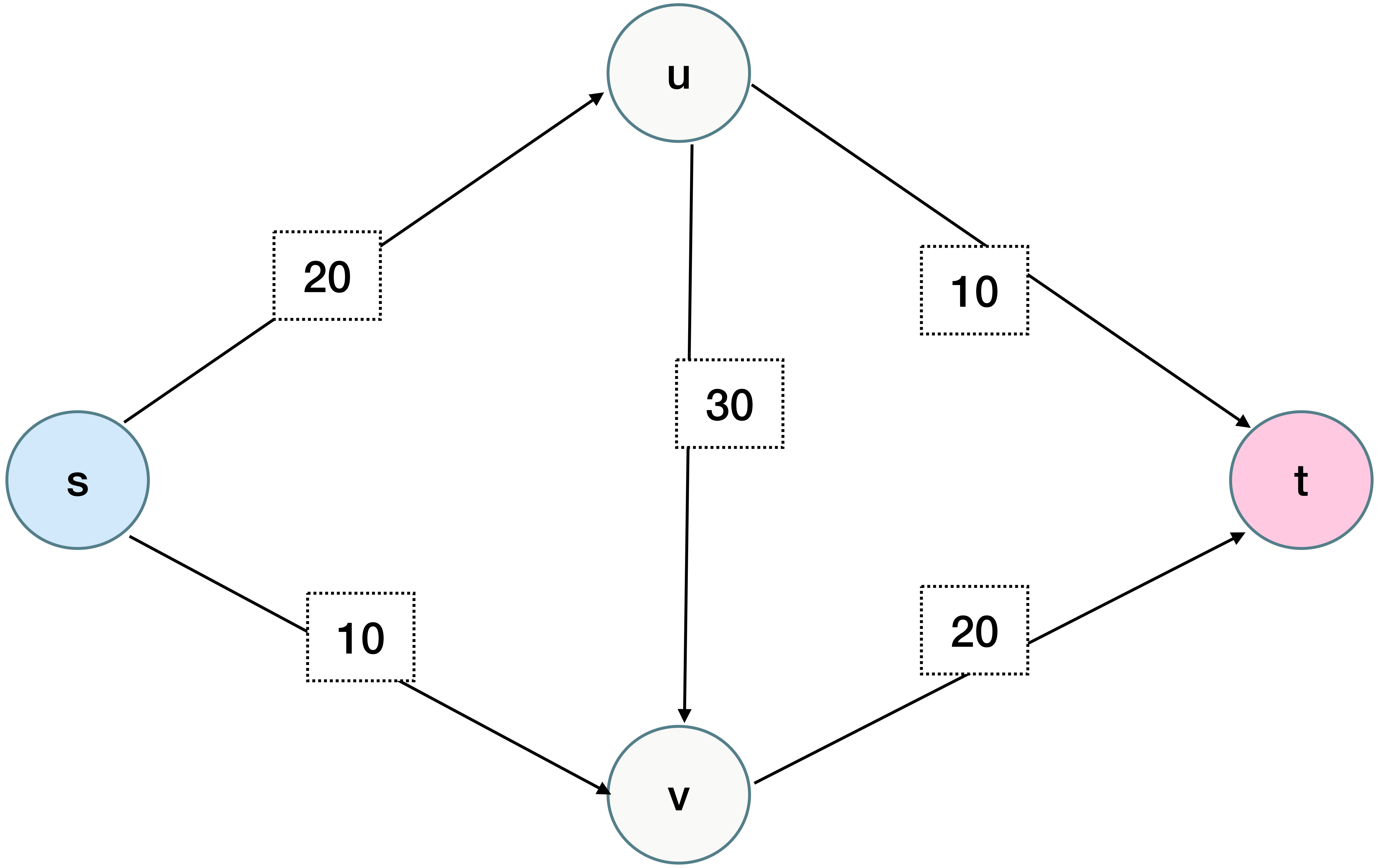
```
augment(f,P)
 let b = bottleneck(P,f)
 for e = (u,v) in P
  if e = (u,v) is forward
    increase f(e) in G by b
  else (u,v) is backward let e = (v,u)
    decrease f(e) in G by b
  endif
 endfor
 return f
```

Recall: Basic Algorithm

We can then find a maximum flow as follows

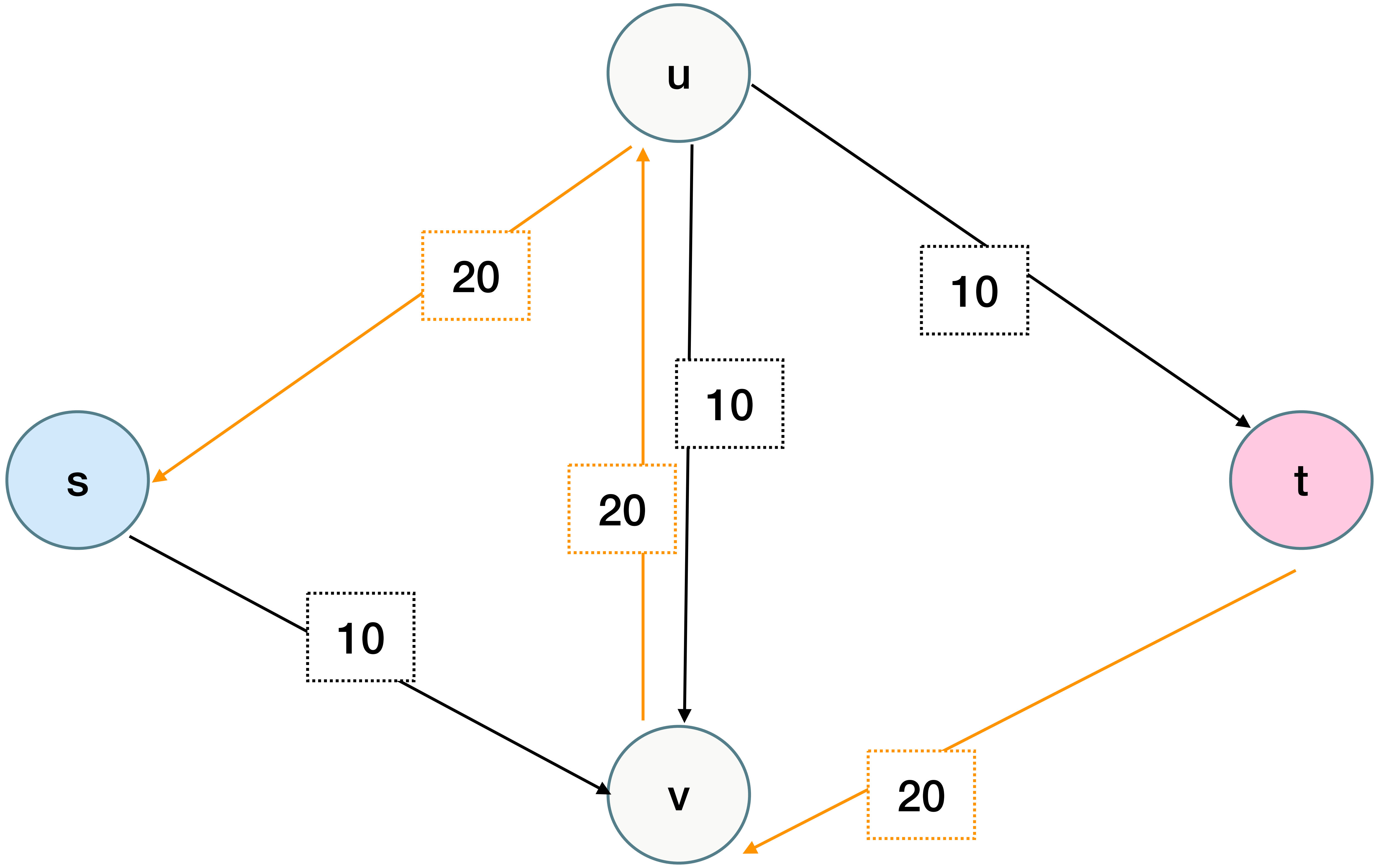
```
MaxFlow (G)
  set f(e) = 0 for all e in G
  while there is an s-t path in $G_f$
     let P be a simple s-t path in $G_f$
      f’ = augment(f, P)
      f = f’
      $G_f$ = $G_{f'}$
  endwhile
 return f
```

Recall: Basic Algorithm



Initially, flow is 0, and $G_f = G$

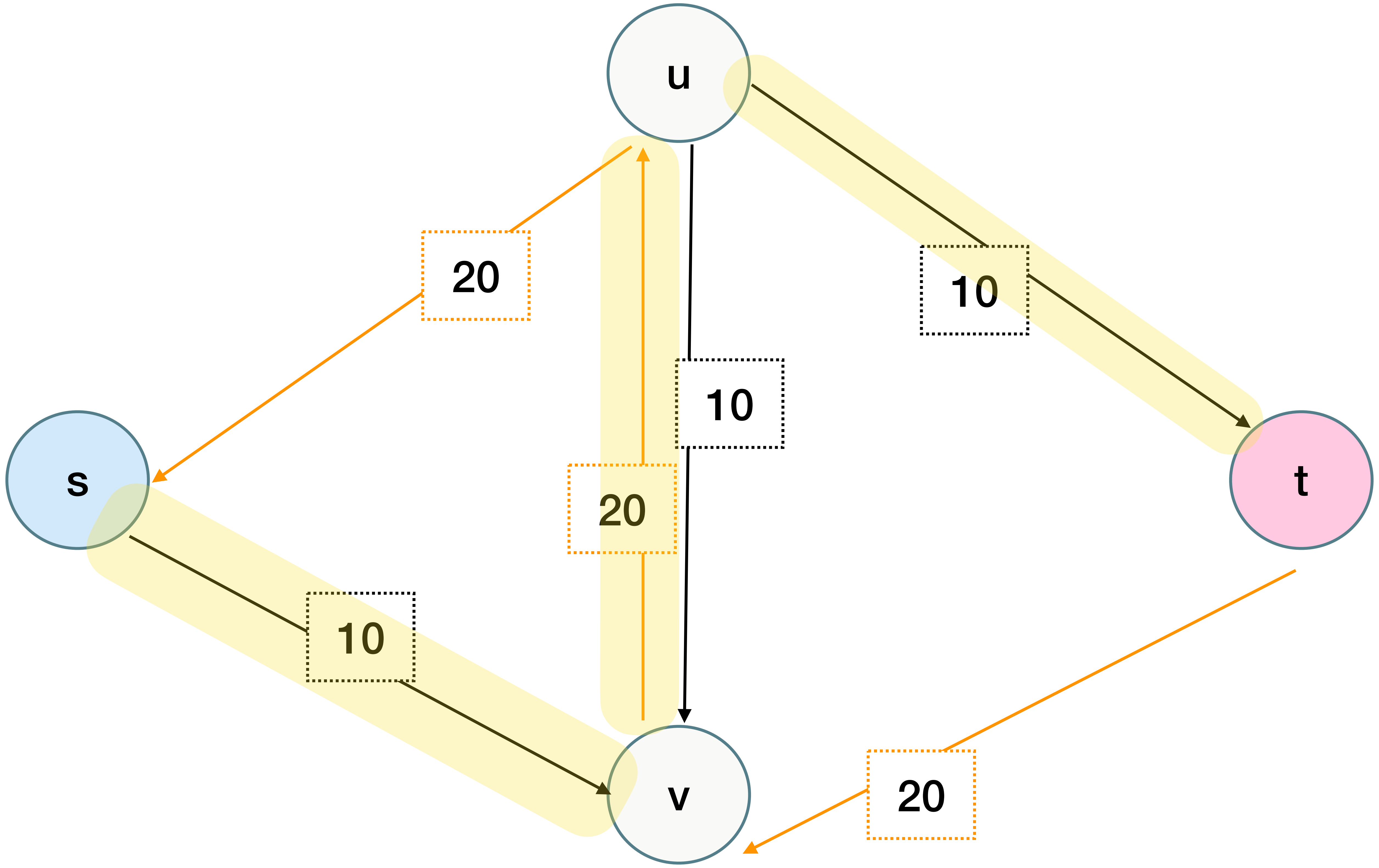
Recall: Basic Algorithm



S -> u -> v -> t is chosen, and we push 20 units across it

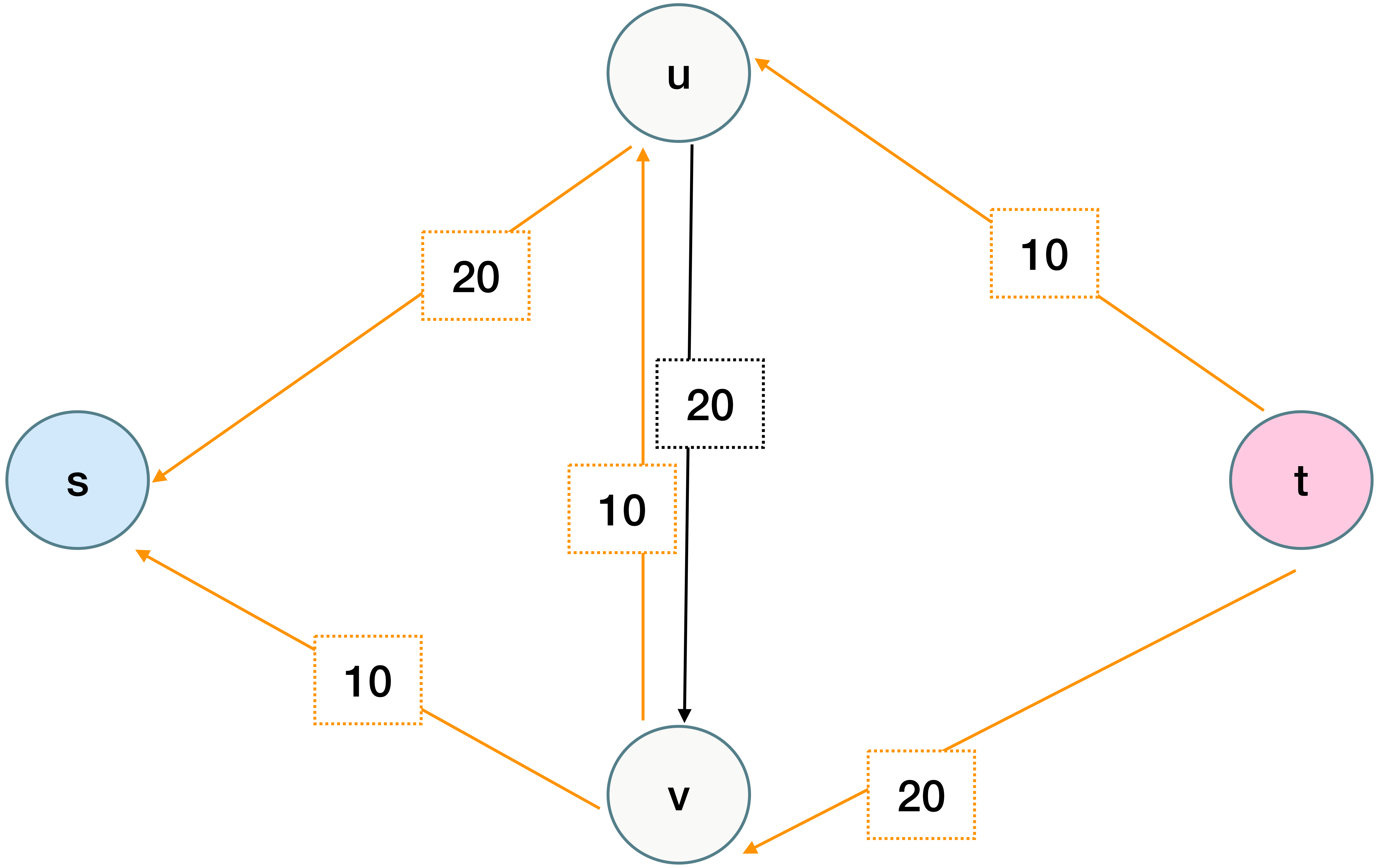
We update G_f

Recall: Basic Algorithm



S -> v -> u -> t is chosen, and we push 10 units across it

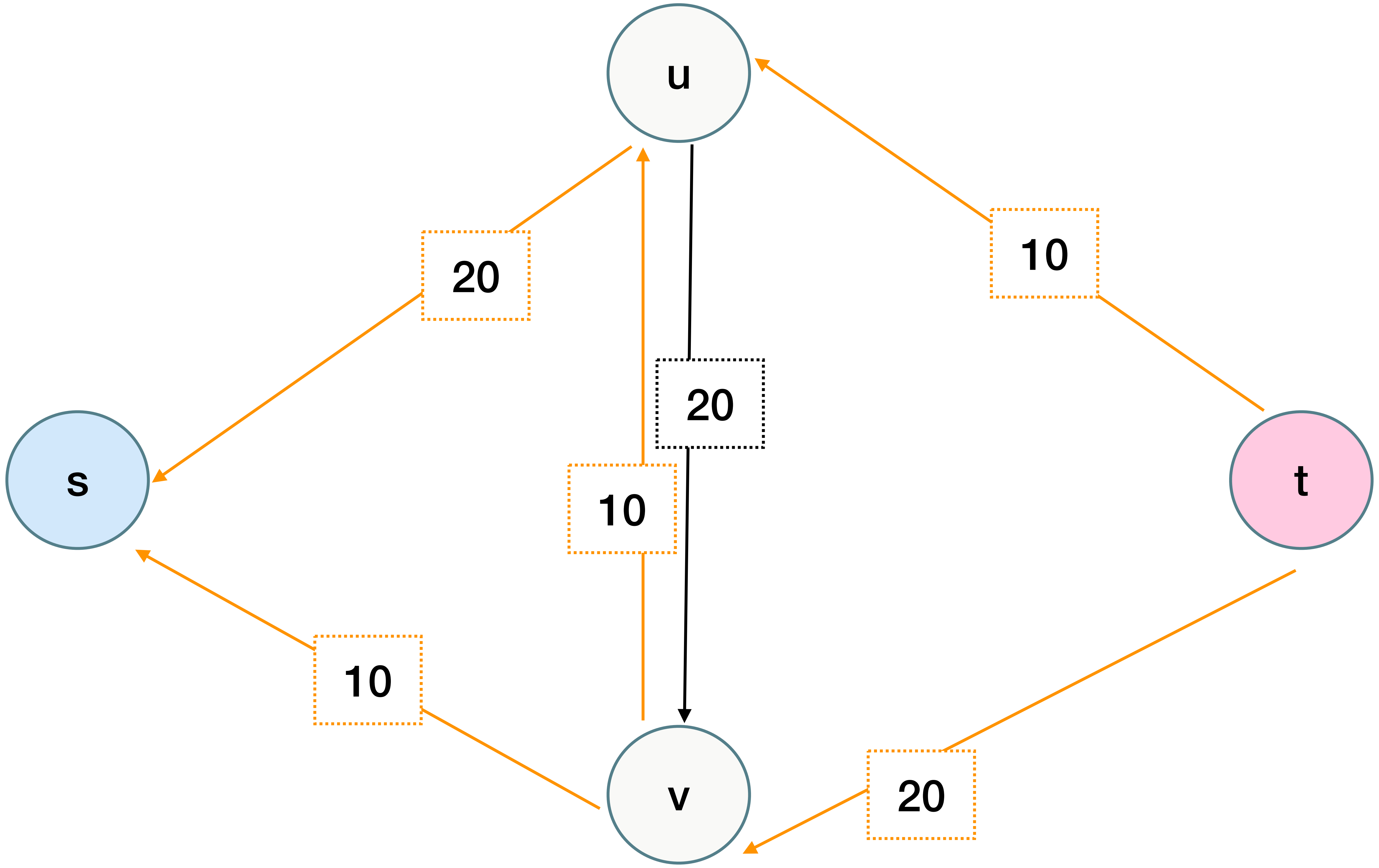
Recall: Basic Algorithm



S -> v -> u -> t is chosen, and we push 10 units across it

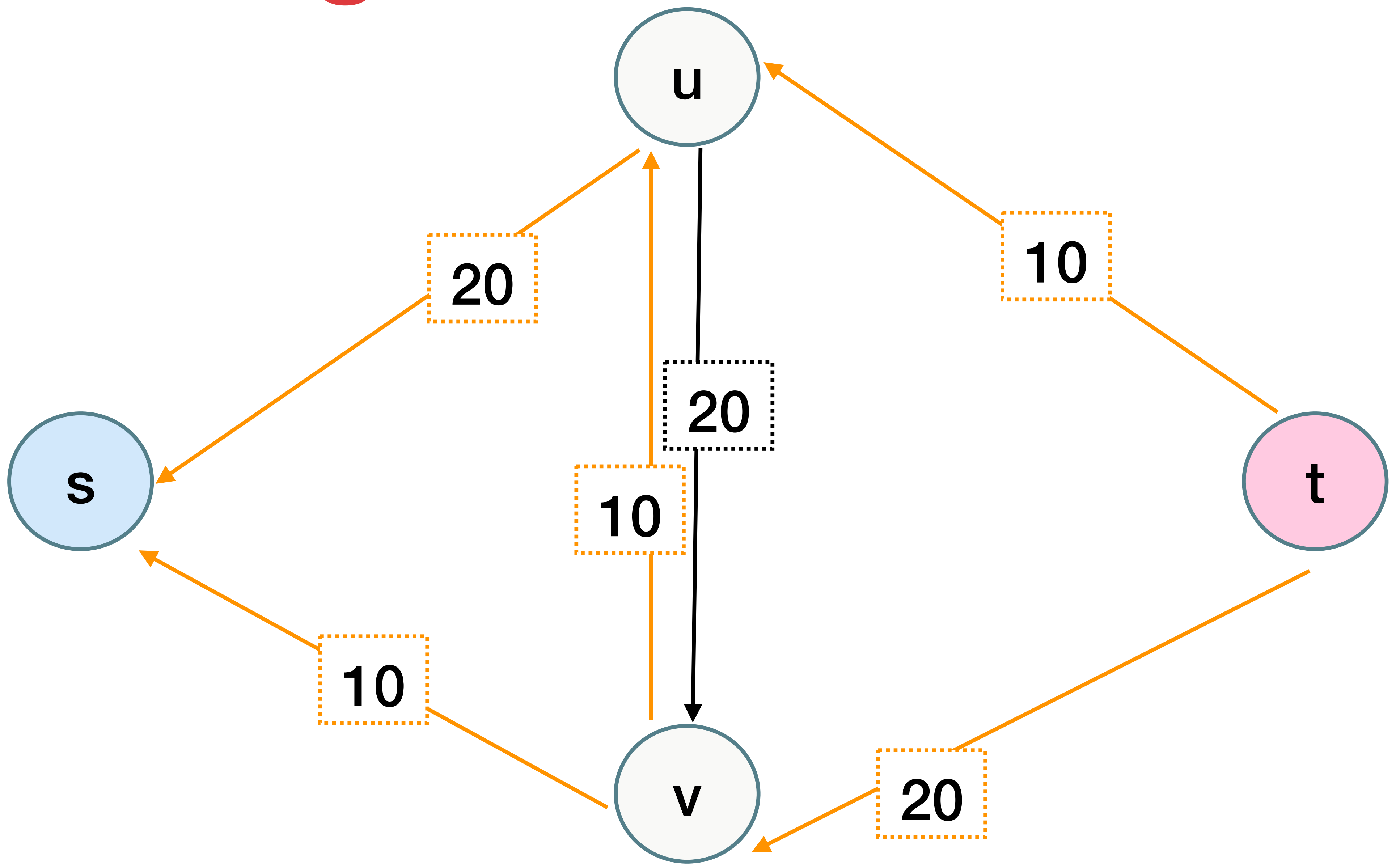
We update G_f

Recall: Basic Algorithm



No more s-t paths in the residual graph. Algorithm terminates with v(f) = 30 (the maximum).

Recall: Basic Algorithm



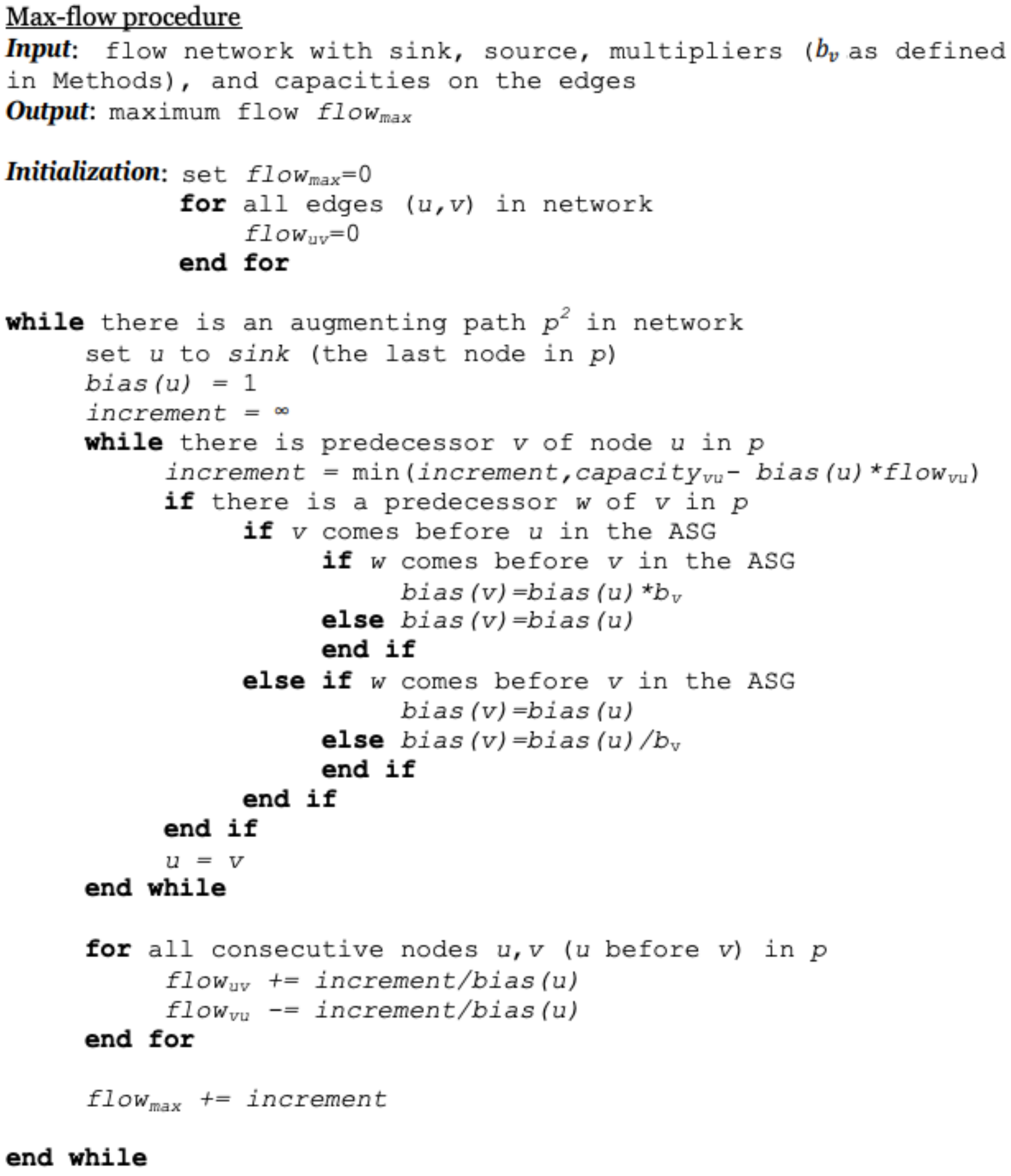
This is the basic Ford-Fulkerson algorithm. Running time is O(mC)

Strongly-polynomial algorithms exist (e.g. Dinitz-Edmonds-Karp $O(nm^2)$)

One can reduce Max Flow with minimum bounds to Max Flow

One can also add gain / loss as is necessary in StringTie

Bias-aware MaxFlow algorithm



Max-flow procedure

Input: flow network with sink, source, multipliers (b_v as defined in Methods), and capacities on the edges

Output: maximum flow $flow_{max}$

```
Initialization: set $flow_{max}$=0
                for all edges ($u$,$v$) in network
                    $flow_{uv}$=0
                end for

while there is an augmenting path $p^2$ in network
     set $u$ to $sink$ (the last node in $p$)
     $bias(u)$ = 1
     $increment$ = $\infty$
     while there is predecessor $v$ of node $u$ in $p$
          $increment$ = min($increment$,$capacity_{vu}$- $bias(u)$*$flow_{vu}$)
          if there is a predecessor $w$ of $v$ in $p$
               if $v$ comes before $u$ in the ASG
                    if $w$ comes before $v$ in the ASG
                         $bias(v)$=$bias(u)$*$b_v$
                    else $bias(v)$=$bias(u)$
                    end if
               else if $w$ comes before $v$ in the ASG
                         $bias(v)$=$bias(u)$
                    else $bias(v)$=$bias(u)$/$b_v$
                    end if
               end if
          end if
          $u$ = $v$
     end while

     for all consecutive nodes $u$,$v$ ($u$ before $v$) in $p$
          $flow_{uv}$ += $increment$/$bias(u)$
          $flow_{vu}$ -= $increment$/$bias(u)$
     end for

     $flow_{max}$ += $increment$

end while
```

Processing the Splice Graph

Repeat:

1. Heuristically choose a “heavy” path (a path with the heaviest node) in the ASG

2. Estimate path expression by computing max-flow in a flow graph corresponding to this sub-path of the ASG. Subtract the read mass assigned to the nodes in this path & repeat.

Until:

Coverage of heaviest path falls below 2.5 reads per-base

Interestingly: Unlike other approaches that try to use the flow graph to ***find*** and ***quantify*** the paths, StringTie uses a heuristic to select the transcript, and flow only to quantify the selected path.

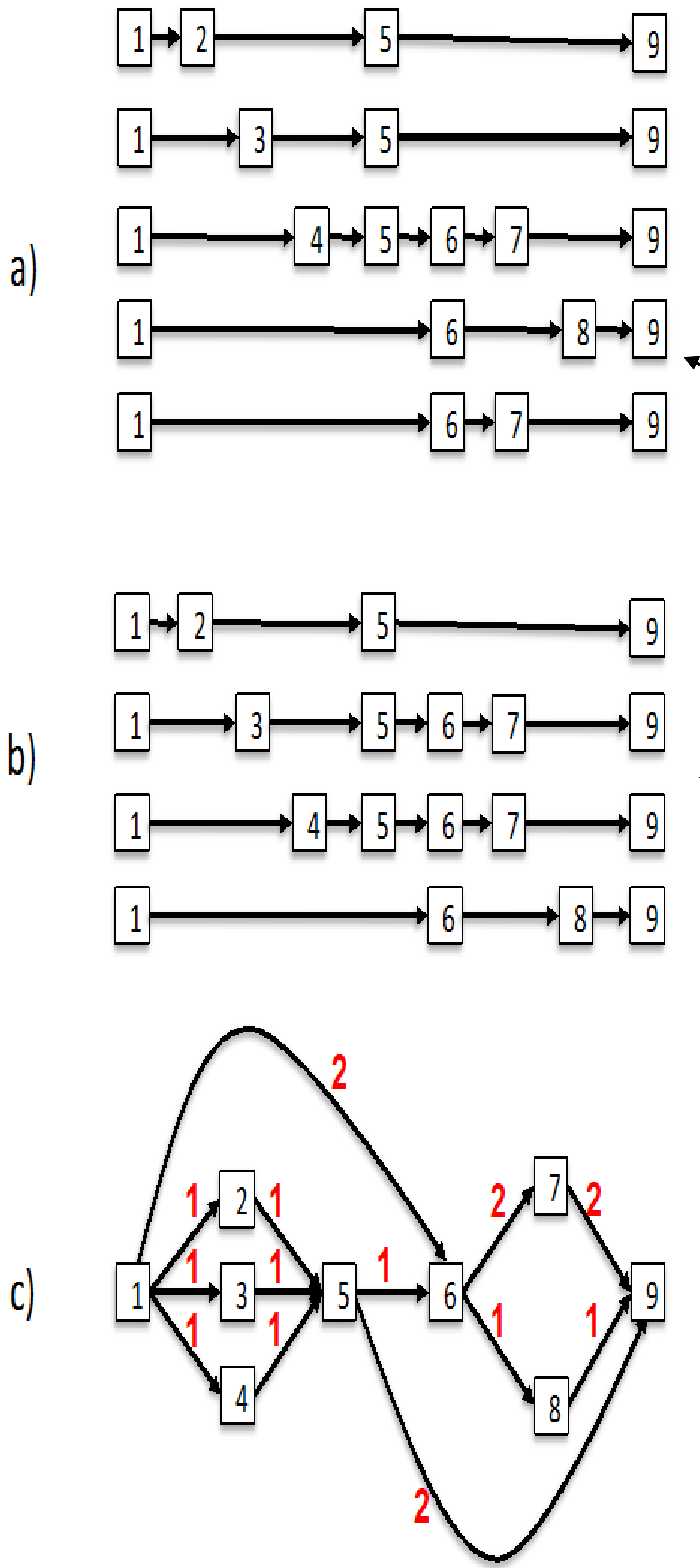


Illustration of how coverage information yields different results from Cufflinks’ “parsimony” approach

transcripts extracted
if we consider coverage

parsimonious set of “covering” transcripts

ASG

Pertea, Mihaela, et al. "StringTie enables improved reconstruction of a transcriptome from RNA-seq reads." *Nature biotechnology* 33.3 (2015): 290.

Results

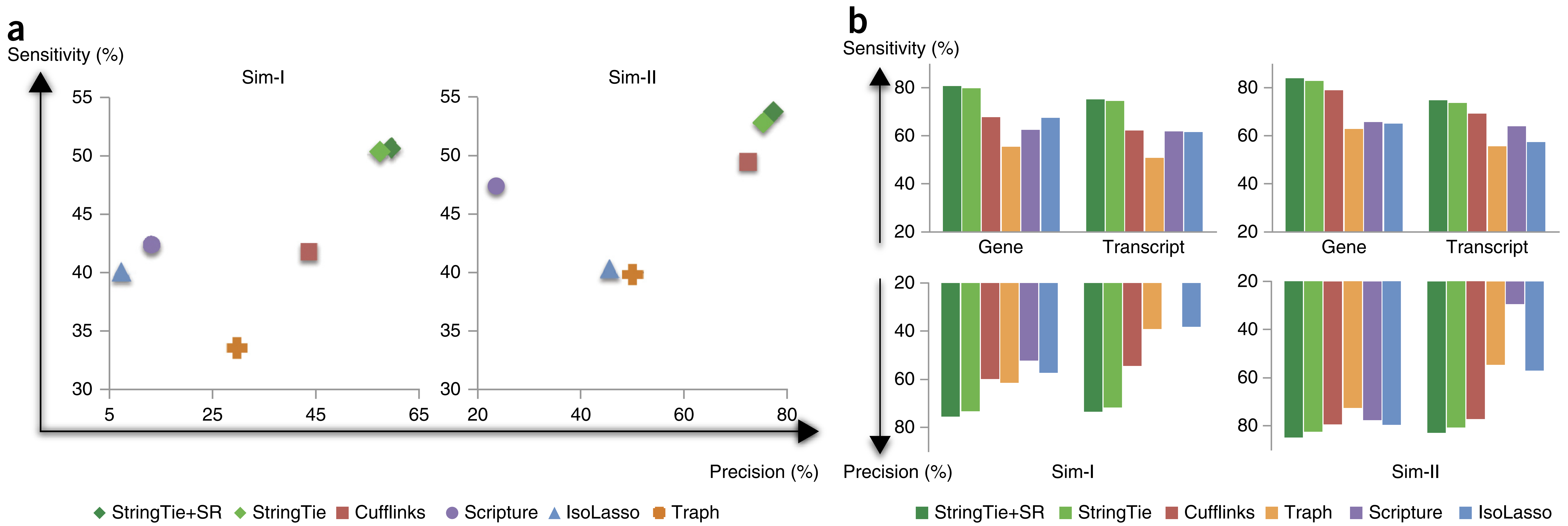


Figure 2 Transcriptome assemblers' accuracies in detecting expressed transcripts from two simulated RNA-seq data sets. (**a**) Transcriptome assemblers' accuracies in detecting expressed transcripts from two simulated RNA-seq data sets. In data set Sim-I (left), the fragment sizes follow an empirical distribution based on Illumina sequences, and in Sim-II (right) the fragment sizes follow a parameterized normal distribution. StringTie+SR pre-assembles the reads into super-reads when possible. (**b**) Accuracy of transcriptome assemblers on gene loci from the same two data sets, considering only those transcripts that were completely covered by input reads. Scripture's precision on Sim-I was 17.7%, below the 20% minimum shown here.

Pertea, Mihaela, et al. "StringTie enables improved reconstruction of a transcriptome from RNA-seq reads." *Nature biotechnology* 33.3 (2015): 290.

Results

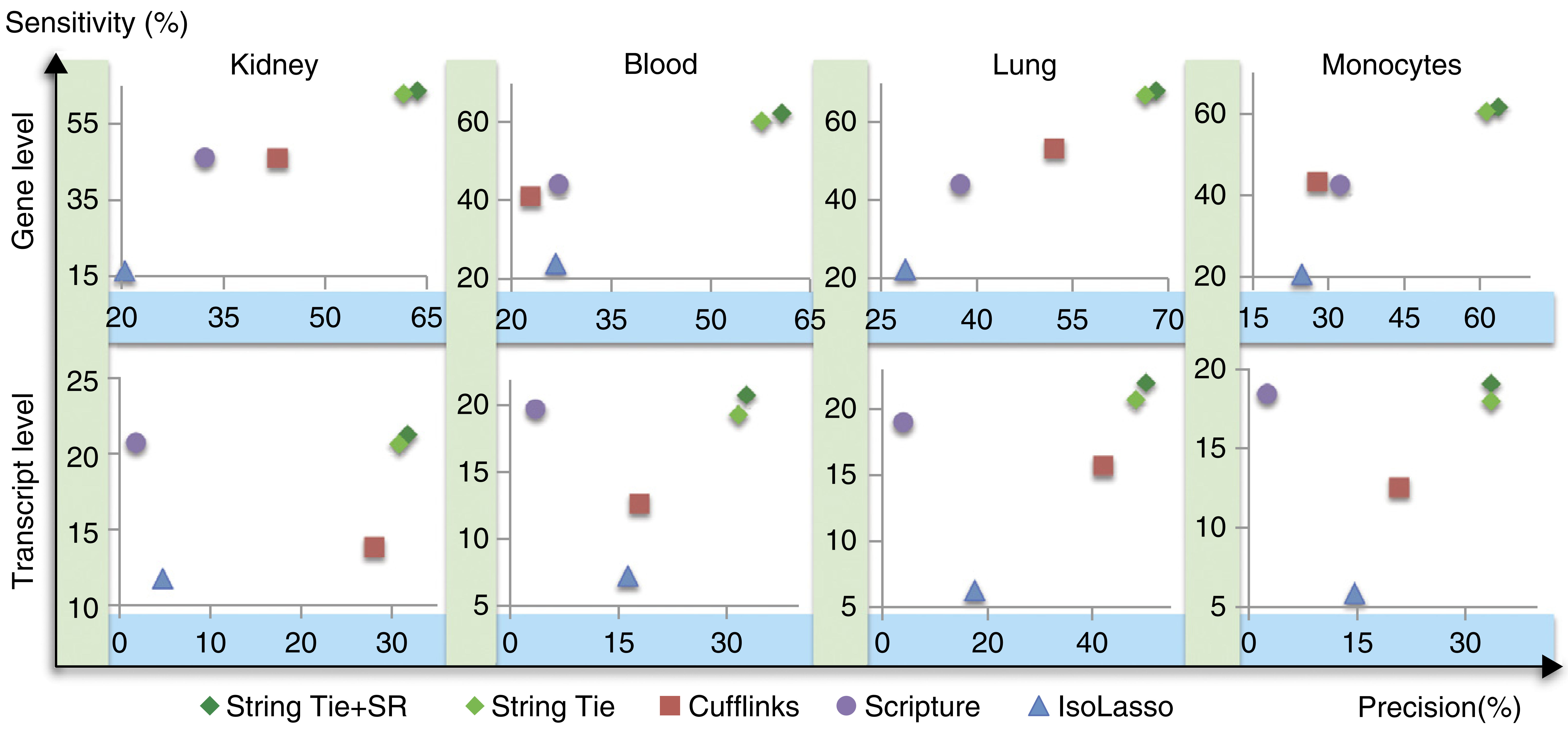
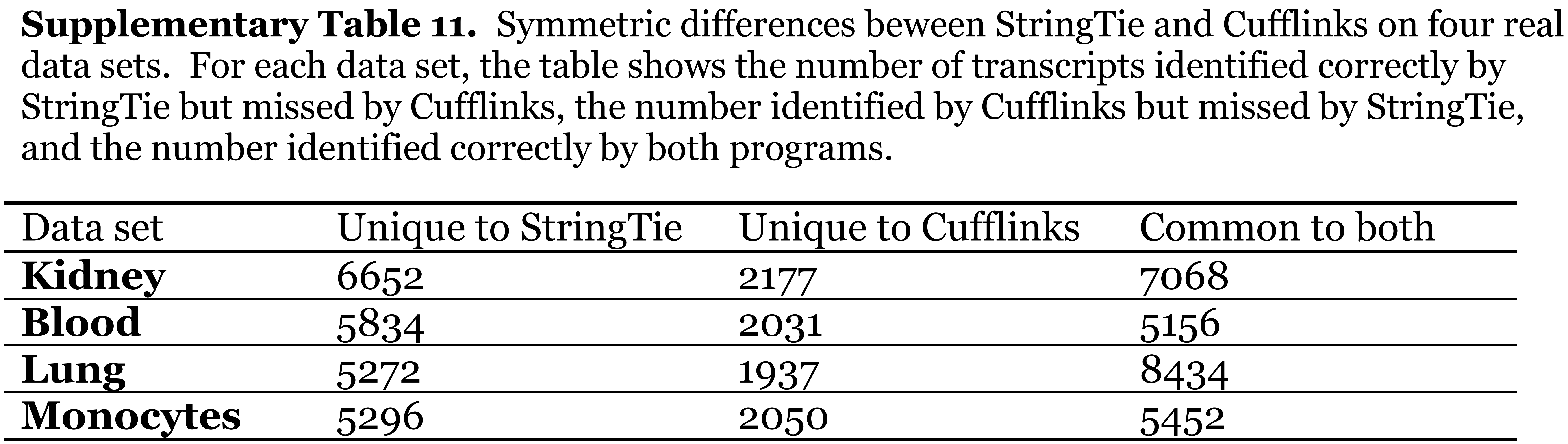


Figure 3 Accuracy of transcript assemblers at assembling known genes, measured on real data sets from four different tissues. Known genes are defined as those annotated in either the RefSeq, UCSC or Ensembl human gene databases. Gene level sensitivity (*y* axis) measures the percentage of genes for which a program got at least one isoform correct, whereas transcript sensitivity measures the percentage of known transcripts that were correctly assembled. Precision (*x* axis) is measured as the percentage of all predicted genes (transcripts) that match an annotated gene (transcript).

Pertea, Mihaela, et al. "StringTie enables improved reconstruction of a transcriptome from RNA-seq reads." *Nature biotechnology* 33.3 (2015): 290.

StringTie Identifies More Transcripts Than Cufflinks

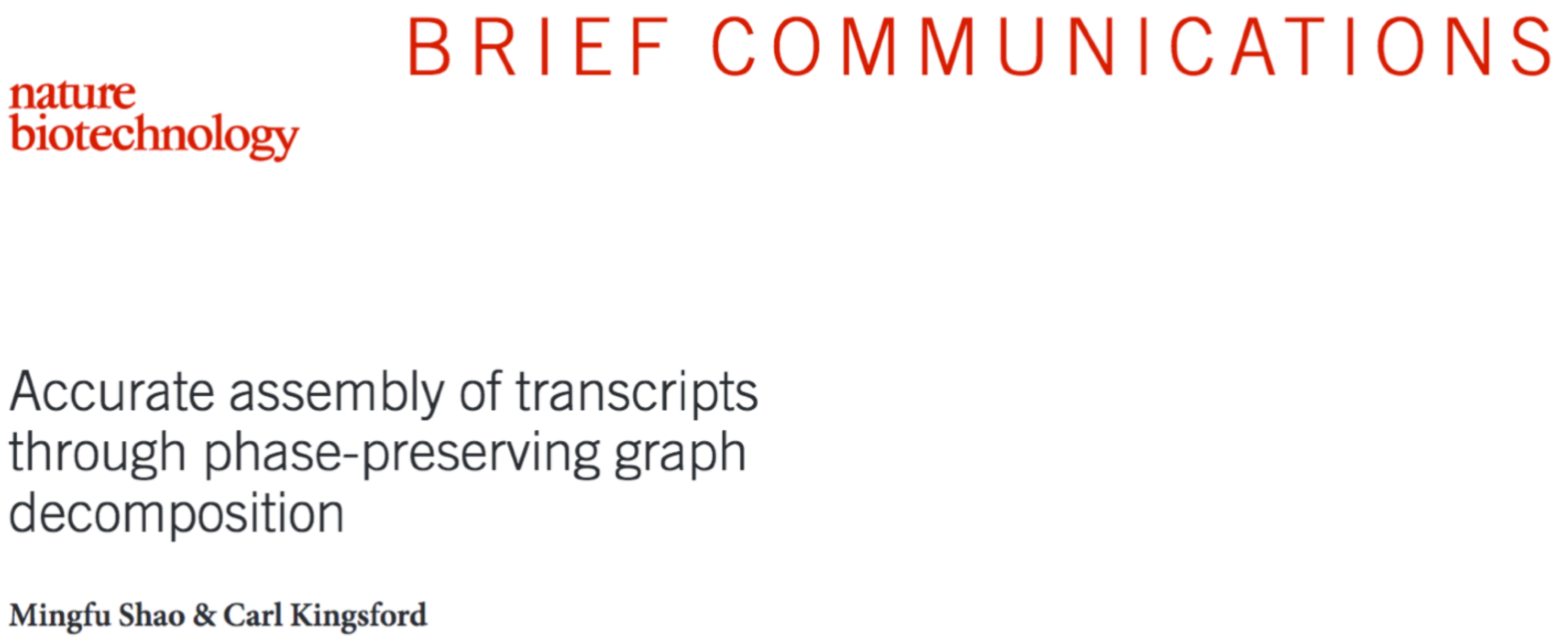


Supplementary Table 11. Symmetric differences beween StringTie and Cufflinks on four real data sets. For each data set, the table shows the number of transcripts identified correctly by StringTie but missed by Cufflinks, the number identified by Cufflinks but missed by StringTie, and the number identified correctly by both programs.

Data set	Unique to StringTie	Unique to Cufflinks	Common to both
Kidney	6652	2177	7068
Blood	5834	2031	5156
Lung	5272	1937	8434
Monocytes	5296	2050	5452

Pertea, Mihaela, et al. "StringTie enables improved reconstruction of a transcriptome from RNA-seq reads." *Nature biotechnology* 33.3 (2015): 290.

Scallop improves assembly by preserving “phasing paths”

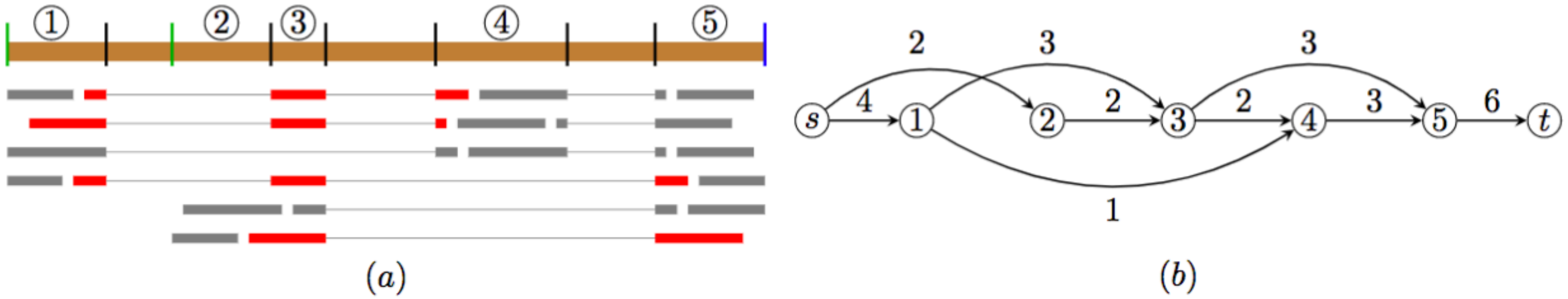


BRIEF COMMUNICATIONS

nature
biotechnology

Accurate assembly of transcripts through phase-preserving graph decomposition

Mingfu Shao & Carl Kingsford



Supplementary Figure 1: Example of building splice graph and phasing paths. **(a)** Alignment of reads to the reference genome. Inferred splice positions are marked with black bars; inferred starting and ending positions are marked with green and blue bars, respectively. Exons and partial exons are labeled with numbers above the reference genome. Reads that span more than two exons are colored red, from which we can get the set of phasing paths as $\{(1,3,4),(2,3,5),(1,3,5)\}$. The abundance of these phasing paths are $g(1,3,4)=2$, $g(2,3,5)=1$, and $g(1,3,5)=1$. **(b)** The corresponding splice graph and weights for all edges.

phasing paths = sub-paths of the splicing graph where read evidence supports >1 splicing junction.

Scallop preserves observed sub-paths.

Shao, Mingfu, and Carl Kingsford. "Accurate assembly of transcripts through phase-preserving graph decomposition." *Nature biotechnology* 35.12 (2017): 1167.

Preserving “phasing paths” improves accuracy

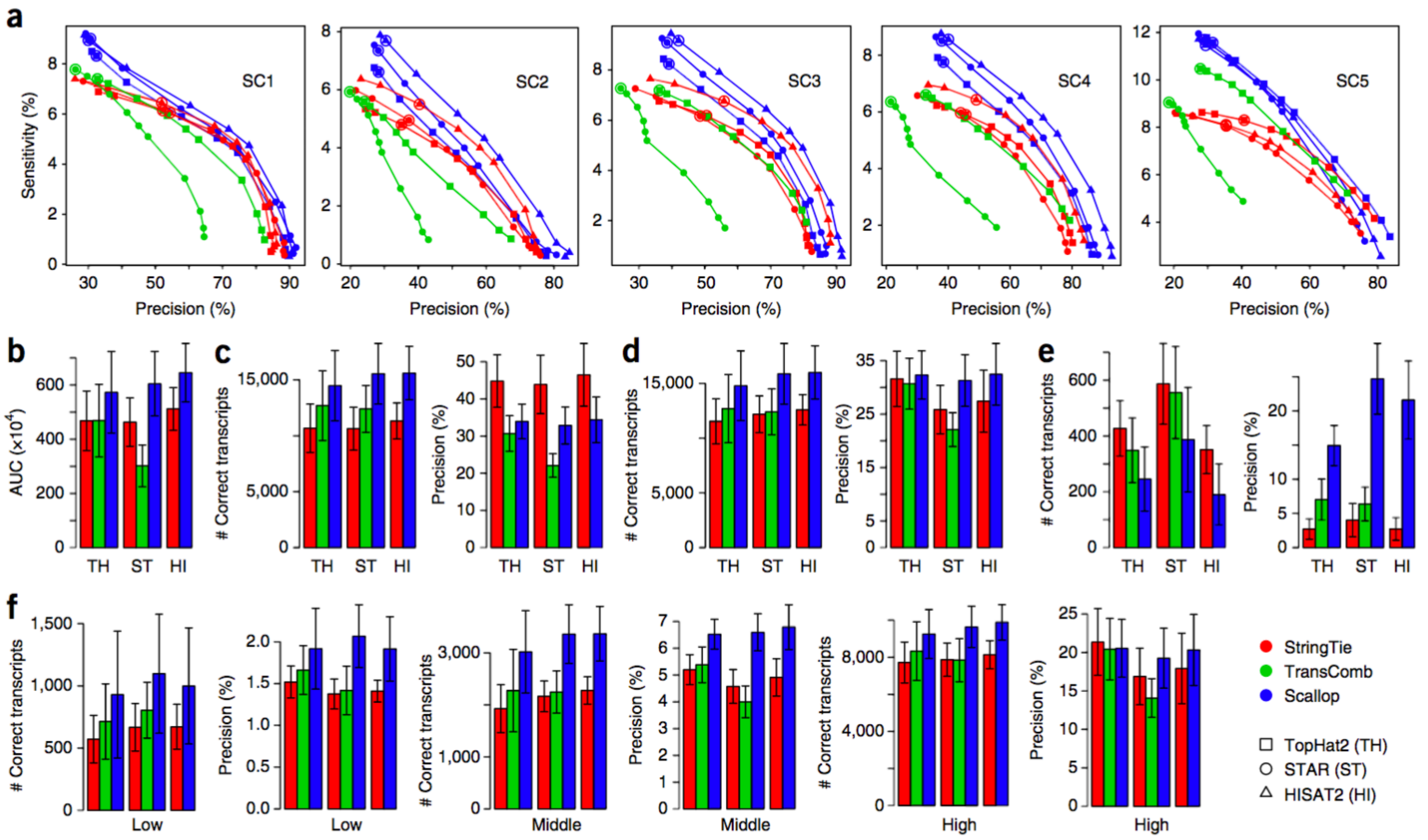
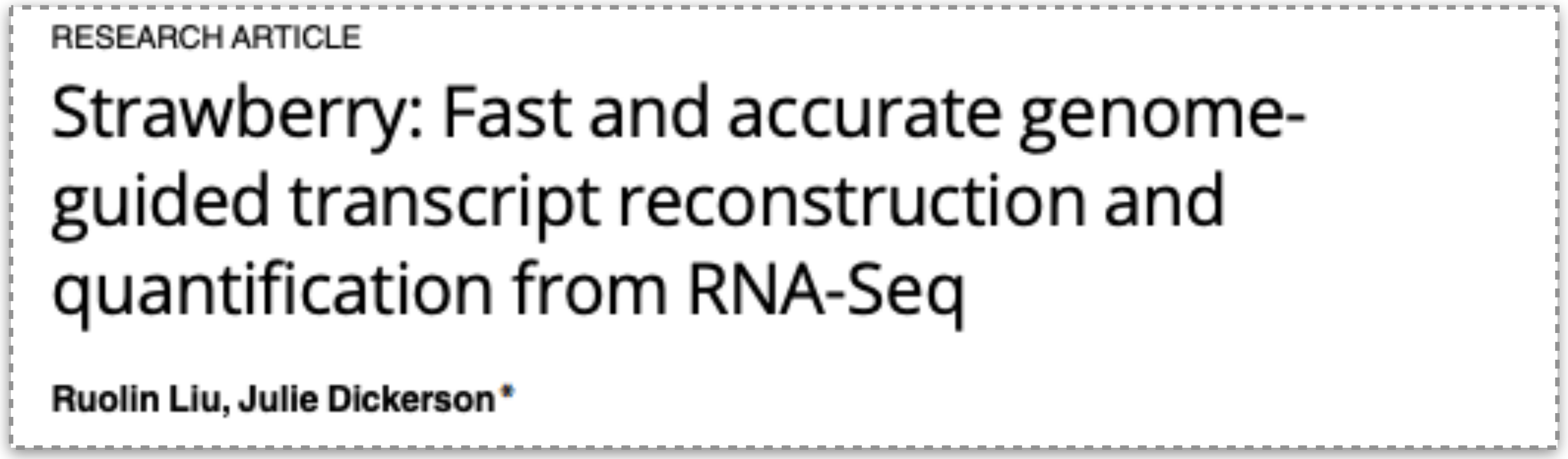


Figure 1 Comparison of the three methods (StringTie, TransComb, and Scallop) over the five testing samples. (**a**) The precision-sensitivity curves for multi-exon transcripts. Each curve connects ten points, corresponding to minimum coverage thresholds {0, 1, 2.5, 5, 7.5, 10, 25, 50, 75, 100}; the default value is circled. (**b**) The average AUC (area under the precision-sensitivity curve) over the five samples. The error bars show the s.d. (the same for other panels). (**c**) The average sensitivity and precision of multi-exon transcripts running at default parameters. (**d**) The average sensitivity and precision of multi-exon transcripts running with minimum coverage set to 0. (**e**) The average sensitivity and precision of single-exon transcripts running at default parameters. (**f**) The average sensitivity and precision of multi-exon transcripts corresponding to low, middle, and high expression levels running with minimum coverage set to 0.

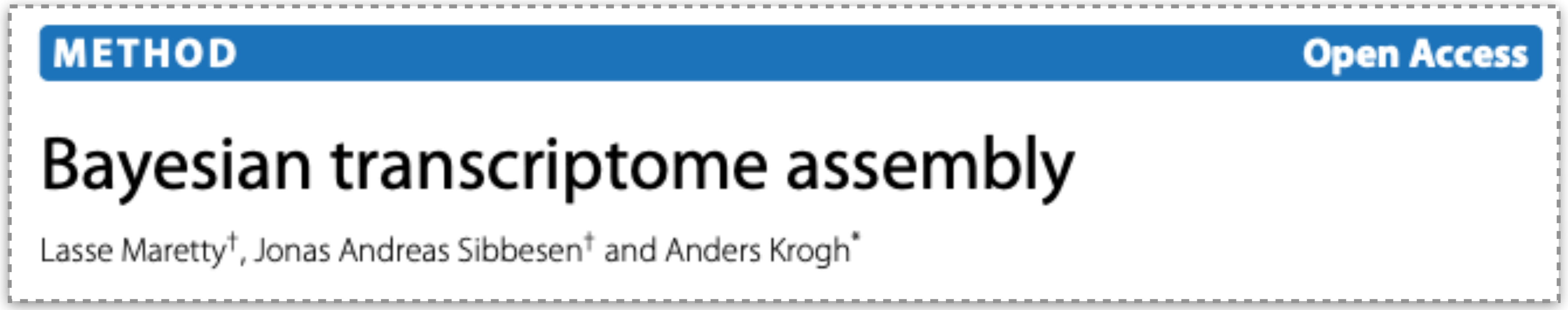
Shao, Mingfu, and Carl Kingsford. "Accurate assembly of transcripts through phase-preserving graph decomposition." *Nature biotechnology* 35.12 (2017): 1167.

Some other interesting assemblers


RESEARCH ARTICLE

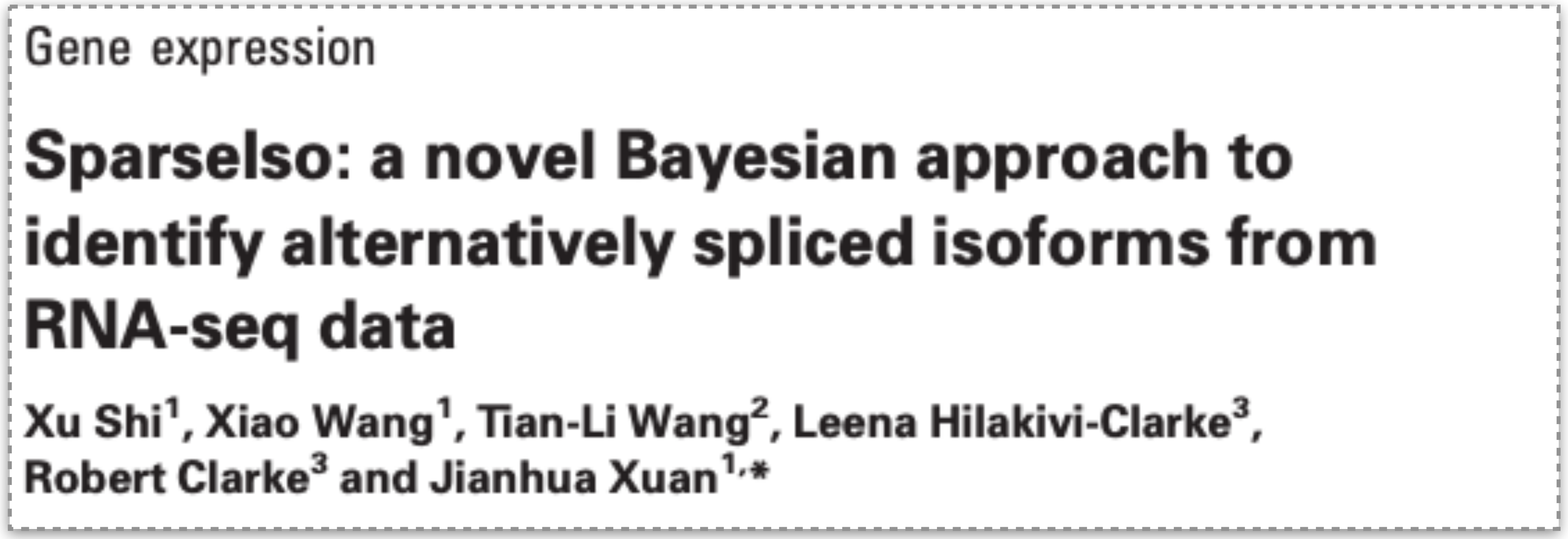
Strawberry: Fast and accurate genome-guided transcript reconstruction and quantification from RNA-Seq

Ruolin Liu, Julie Dickerson*


METHOD Open Access

Bayesian transcriptome assembly

Lasse Maretty[†], Jonas Andreas Sibbesen[†] and Anders Krogh[*]


Gene expression

SparseIso: a novel Bayesian approach to identify alternatively spliced isoforms from RNA-seq data

Xu Shi[1], Xiao Wang[1], Tian-Li Wang[2], Leena Hilakivi-Clarke[3], Robert Clarke[3] and Jianhua Xuan[1,*]

These methods, in particular, solve identification and quantification simultaneously.

Conceptually, this seems like the strongest approach
given how related the problems are.