Computing optimal flow decompositions for assembly

Kyle Kloster, **Philipp Kuinke**, Michael P. O'Brien, Felix Reidl, Fernando Sánchez Villaamil, Blair D. Sullivan, Andrew van der Poel

2018/03/27

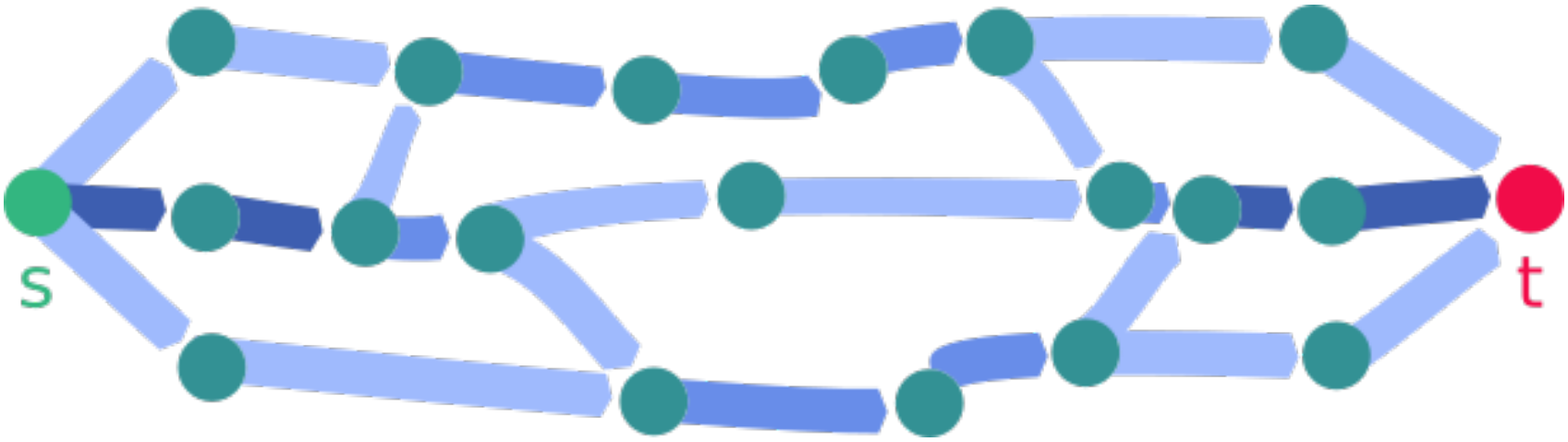
North Carolina State University
RWTH Aachen University

Motivation

The Problem

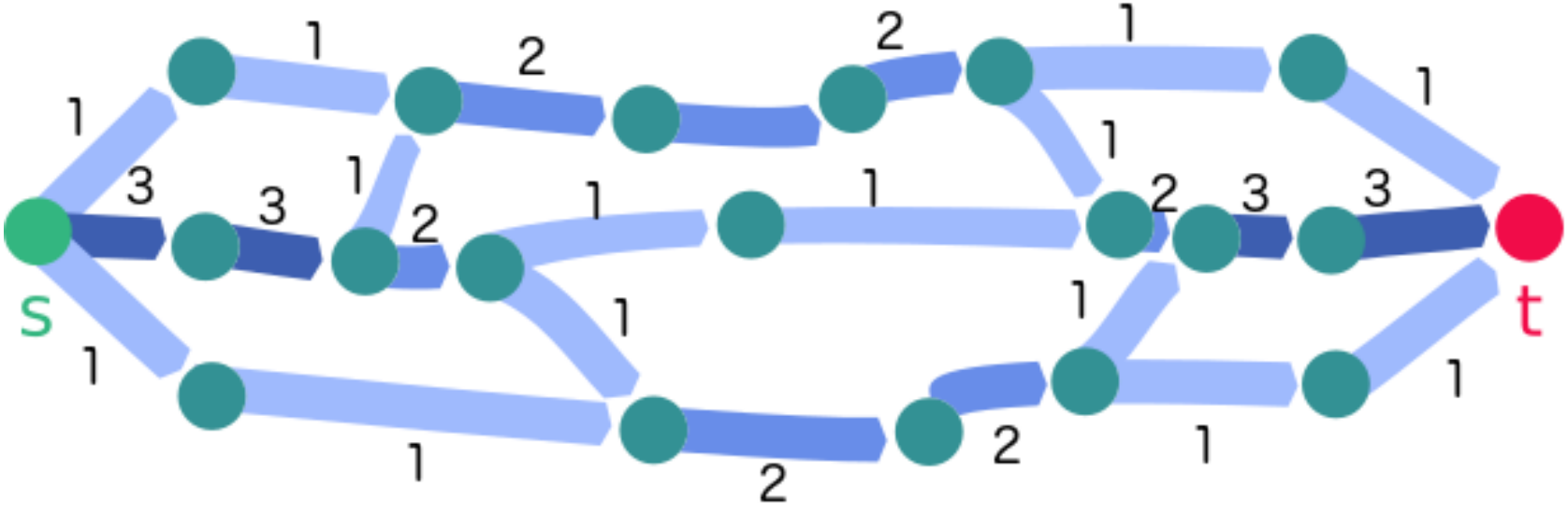
Shared segments between DNA/RNA strands create ambiguity in the assembly problem

The Problem

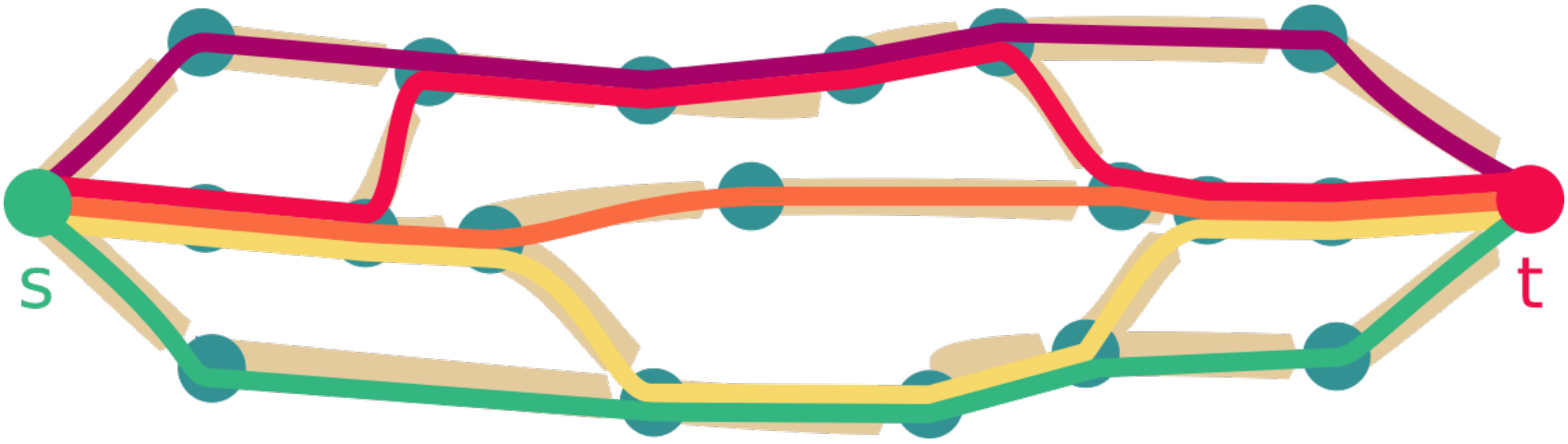


Connecting overlapping segments and counting their frequencies yields a splice-graph.

The Problem

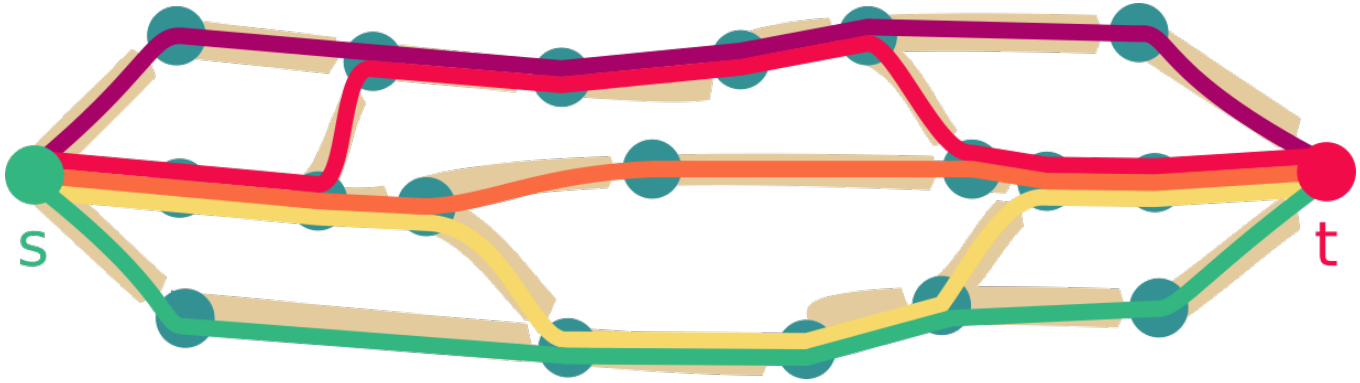


The Problem



The problem is to split the flow into s-t-paths, to recover the original DNA/RNA strands.

The Problem



AGGACGTAGATAGCTAGCTAATGCTACGATCAGAGGACGTAGATTTATTACCAT

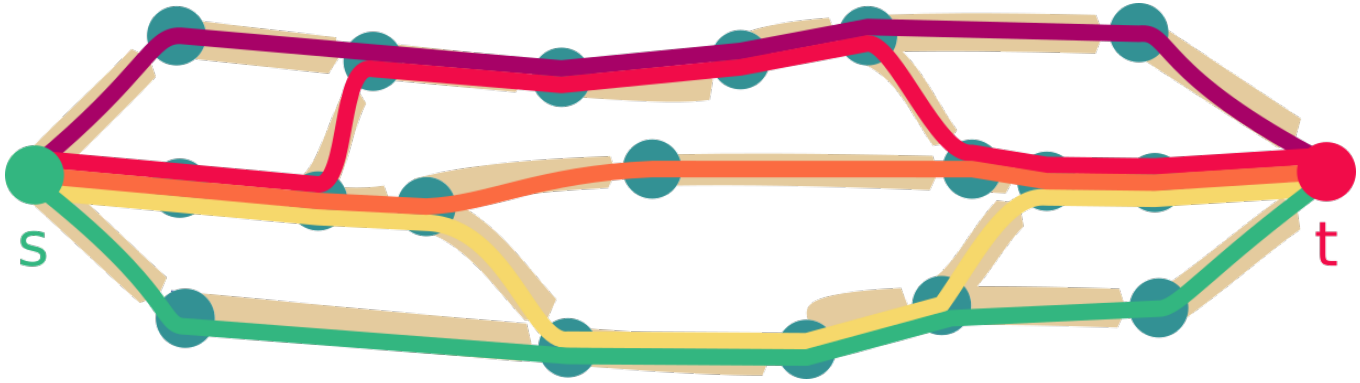
TACCGAATACGAACTAGGATATCGATCGATCAGAGGCCCAATAGGGAATATCCG

TACCGAATACGAACTAGGATATCGATCGATTGATCTATAATAGTAGAATATCCG

AGGACGTAGATAGCTAGCTAATGCTACGATCAGAGGACGTAGATTTATTACCAT

TACCGAATACGAACTAGGATATCGATCGATCAGAGGCCCAATAGGGAATATCCG

The Problem



k-Flow Decomposition (*k*-FD)

Input: (G, f, k) with an s-t–DAG G, a flow f on G, and a positive integer k.

Problem: Find an integral *flow decomposition* of (G, f) using at most k paths.

Related Work

k-Flow Decomposition (*k*-FD)

Input: (G, f, k) with an s-t–DAG G, a flow f on G, and a positive integer k.

Problem: Find an integral *flow decomposition* of (G, f) using at most k paths.

Related Work

k-Flow Decomposition (k-FD)

Input: (G, f, k) with an s-t–DAG G, a flow f on G, and a positive integer k.

Problem: Find an integral *flow decomposition* of (G, f) using at most k paths.

How do we choose k?

Related Work

k-Flow Decomposition (k-FD)

Input: (G, f, k) with an s-t–DAG G, a flow f on G, and a positive integer k.

Problem: Find an integral *flow decomposition* of (G, f) using at most k paths.

How do we choose k?
$\rightarrow$ minimization

A novel min-cost flow method for estimating transcript expression with RNA-Seq.
A.I. Tomescu et. al.

Efficient Heuristic for Decomposing a Flow with Minimum Number of Paths.
M. Shao & C. Kingsford

Related Work

k-Flow Decomposition (k-FD)

Input: (G, f, k) with an s-t–DAG G, a flow f on G, and a positive integer k.

Problem: Find an integral *flow decomposition* of (G, f) using at most k paths.

How do we choose k?
$\rightarrow$ minimization

A novel min-cost flow method for estimating transcript expression with RNA-Seq.
A.I. Tomescu et. al.

Efficient Heuristic for Decomposing a Flow with Minimum Number of Paths.
M. Shao & C. Kingsford

Problem is NP-hard even for weights $\{1, 2, 4\}$

How to split a flow?
T. Hartman et. al.

Computer Scientists…

About ten years ago, some computer scientists came by and said they heard we have some really cool problems. They showed that the problems are NP-complete and went away!

-Joseph Felsenstein (Biologist)

Computer Scientists…

About ten years ago, some computer scientists came by and said they heard we have some really cool problems. They showed that the problems are NP-complete and went away!

-Joseph Felsenstein (Biologist)

Linear FPT

$$c^n \quad \text{vs.} \quad d^k \cdot n$$

linear fpt: exponential only in the *parameter* and linear in n!

Observations

Data used by Shao and Kingsford:

1. 99% of instances decompose into ≤ 8 paths.
   $\rightarrow$ exploit **small natural parameter**.
2. ~4 million mostly small instances.
   $\rightarrow$ handle **large throughput**.
3. Output decompositions.
   $\rightarrow$ reliably recover **domain-specific solution**.

Toboggan

Theorem

Toboggan *solves* k-FD *in* $2^{O(k^2)}(n + \lambda)$*, where* λ *is the logarithm of the largest flow value.*

Toboggan

Theorem

Toboggan *solves* k-FD *in* $2^{O(k^2)}(n + \lambda)$*, where* λ *is the logarithm of the largest flow value.*

- Worst-case run-time is linear in n

Toboggan

Theorem

Toboggan *solves* k-FD *in* $2^{O(k^2)}(n + \lambda)$*, where* λ *is the logarithm of the largest flow value.*

- Worst-case run-time is linear in n
- Guarantees optimal solution

Toboggan

Theorem

Toboggan *solves* k-FD *in* $2^{O(k^2)}(n + \lambda)$*, where* λ *is the logarithm of the largest flow value.*

- Worst-case run-time is linear in n
- Guarantees optimal solution
  - **Gives opportunity to validate the model**

Toboggan

Theorem

Toboggan *solves* k-FD *in* $2^{O(k^2)}(n + \lambda)$*, where* λ *is the logarithm of the largest flow value.*

- Worst-case run-time is linear in n
- Guarantees optimal solution
  - **Gives opportunity to validate the model**
- Run-time competitive with current state of the art heuristic

Toboggan

Theorem

Toboggan *solves* k-FD *in* $2^{O(k^2)}(n + \lambda)$*, where* λ *is the logarithm of the largest flow value.*

- Worst-case run-time is linear in n
- Guarantees optimal solution
  - **Gives opportunity to validate the model**
- Run-time competitive with current state of the art heuristic
- Usable in practice

Implementation & Experiments

Repository

https://github.com/theoryinpractice/toboggan

Setup

Dataset: Available from Shao and Kingsford. Simulated sequencing data for `human`, `mouse` and `zebrafish`, containing ground-truth.

Setup

Dataset: Available from Shao and Kingsford.
Simulated sequencing data for `human`, `mouse` and `zebrafish`, containing ground-truth.

Deviation from original setup:
Trivial instances omitted.
Removes around 64% of the 4M graphs.

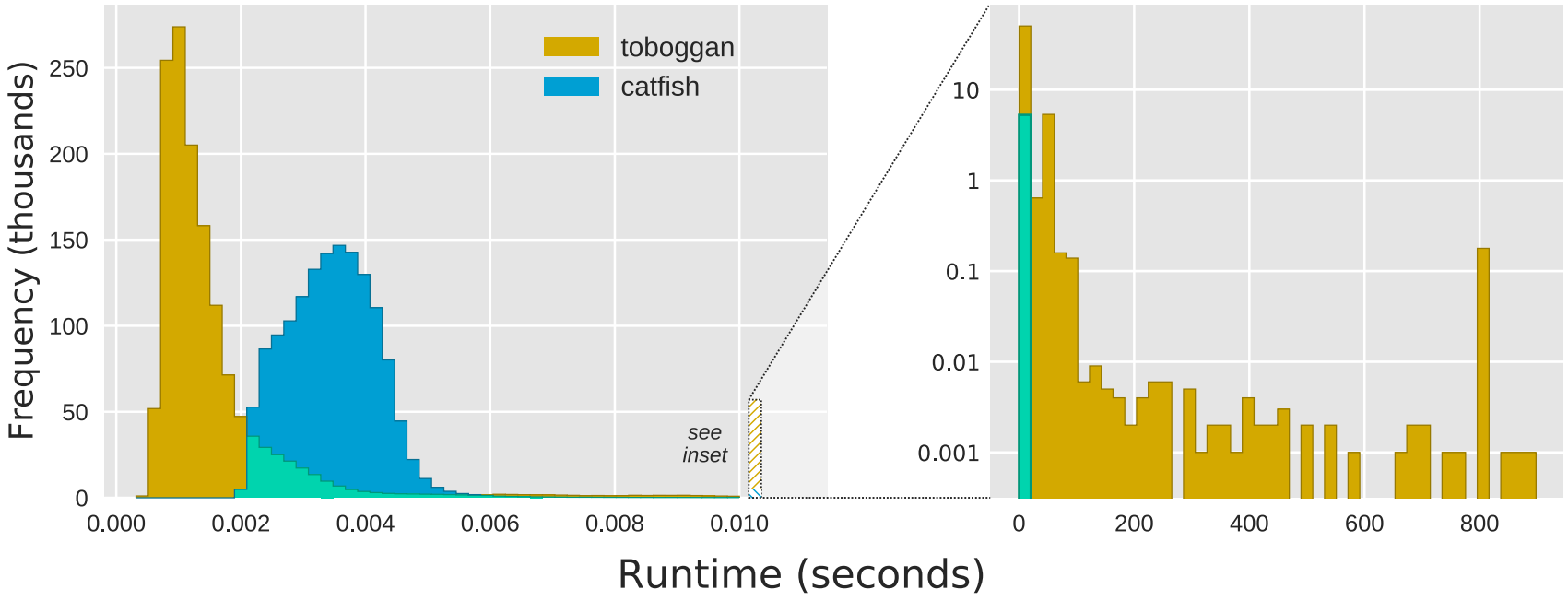
Setup

Dataset: Available from Shao and Kingsford.
Simulated sequencing data for `human`, `mouse` and `zebrafish`, containing ground-truth.

Deviation from original setup:
Trivial instances omitted.
Removes around 64% of the 4M graphs.

Dedicated system with Intel i7-3770:
3.40 GHz, 8 MB cache and 32 GB RAM.

Execution Time



Median:

Toboggan: 1.24ms

Catfish: 3.47ms

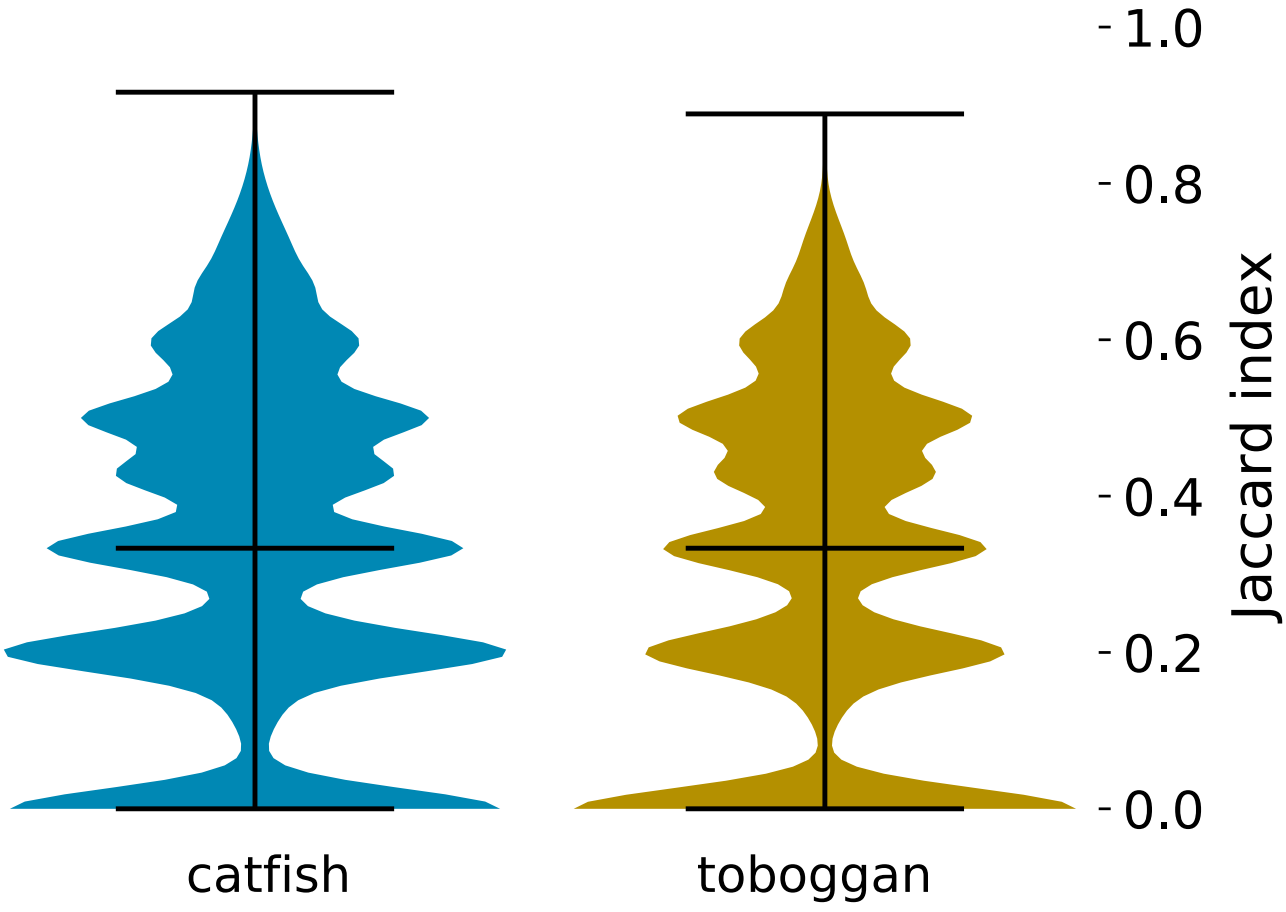
Ground Truth Validation

dataset	instances	minimal	non-minimal
`zebrafish`	445,880	99.907%	0.053%
`mouse`	473,185	99.401%	0.074%
`human`	529,523	99.490%	0.043%
all	1,448,588	99.589%	0.056%

Exact Recovery

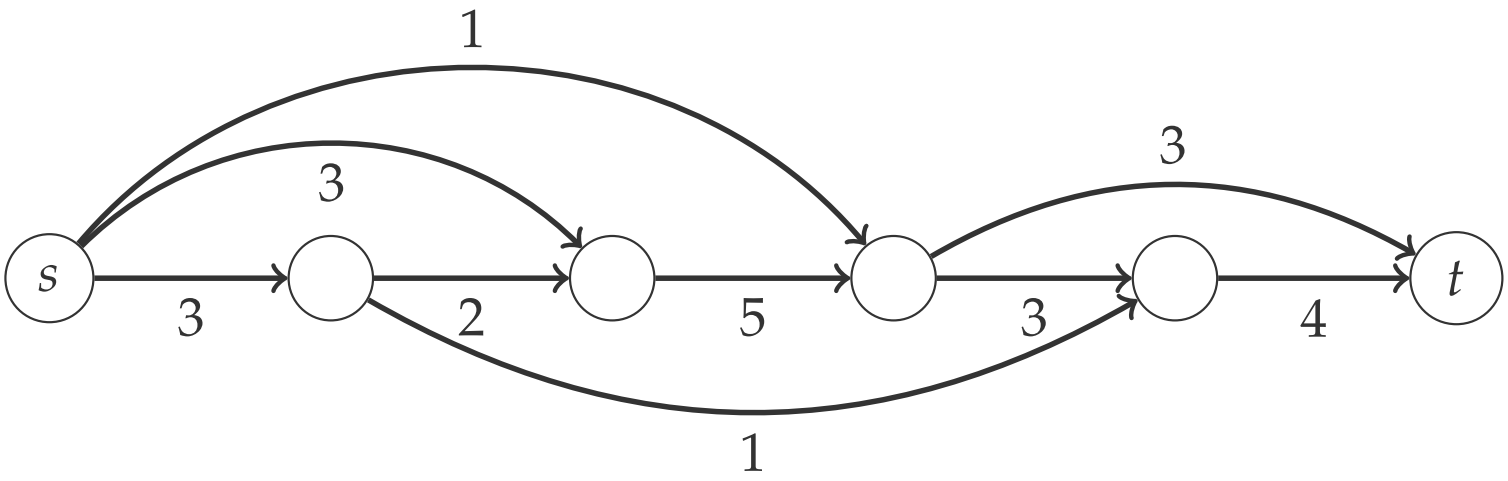
k	instances	Catfish	Toboggan
2	63.2791%	0.992	**0.995**
3	22.0775%	0.967	**0.969**
4	8.5237%	**0.931**	0.930
5	3.4920%	0.886	0.886
6	1.5375%	**0.830**	0.828
7	0.6698%	**0.788**	0.780
8	0.2889%	**0.767**	0.766
9	0.1241%	0.740	**0.743**
10	0.0070%	0.752	**0.802**
11	0.0004%	0.500	0.500
all	100%	0.973	**0.975**

Solutions vs. Ground Truth

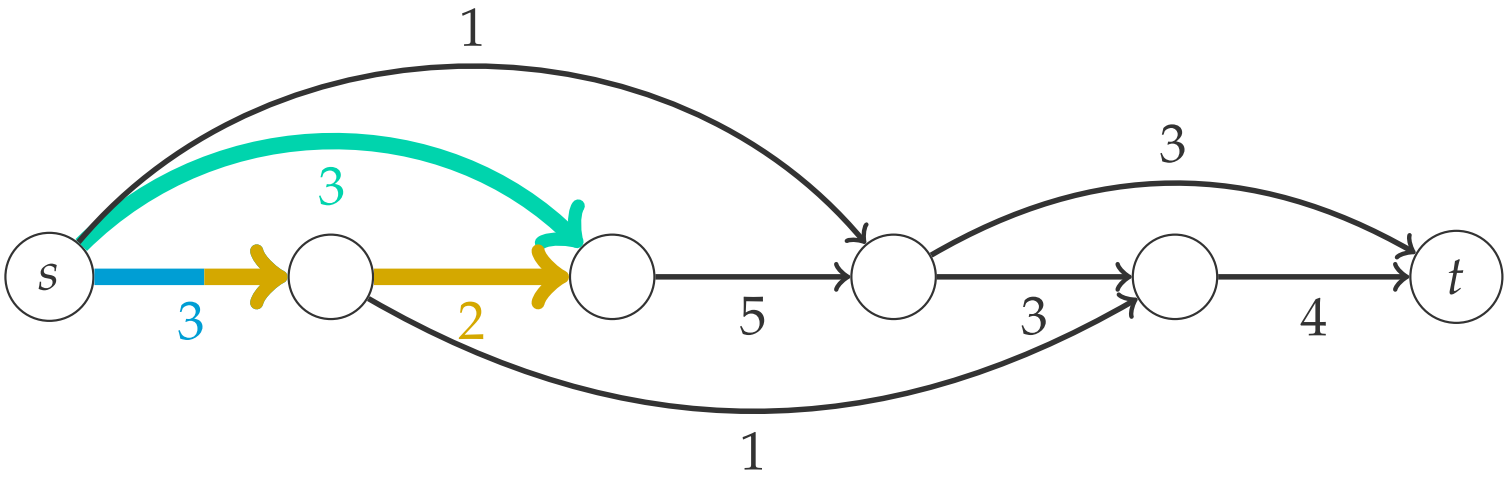


Algorithm Idea

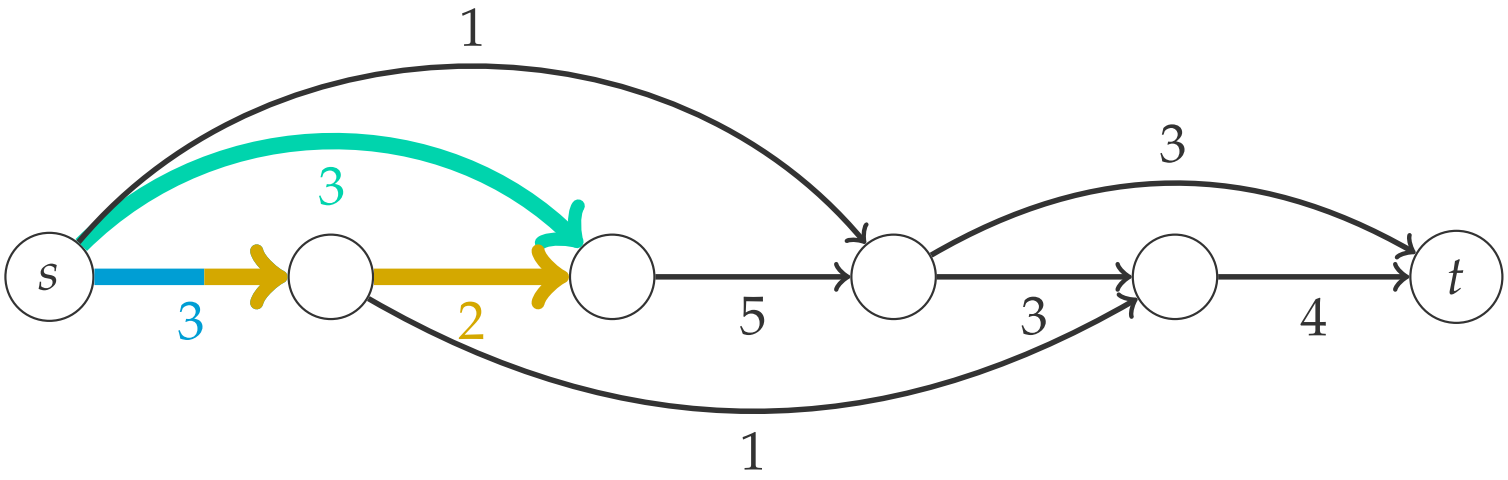
The Idea



The Idea



The Idea

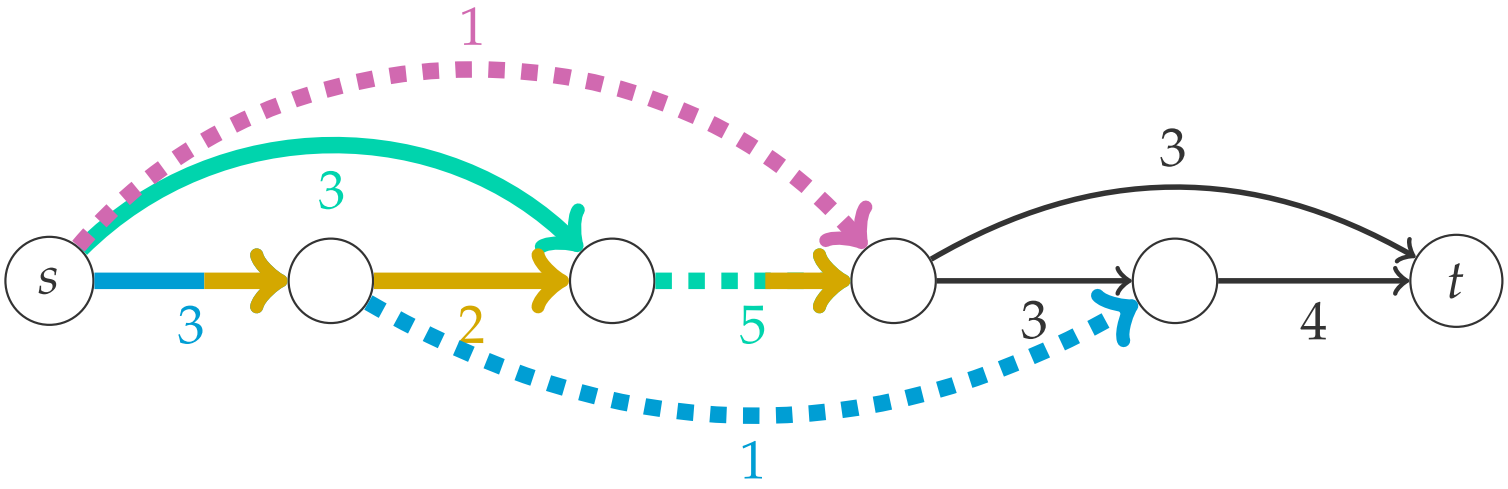


$$w_{\blacksquare} + w_{\blacksquare} = 3$$

$$w_{\blacksquare} = 3$$

$$w_{\blacksquare} = 2$$

The Idea

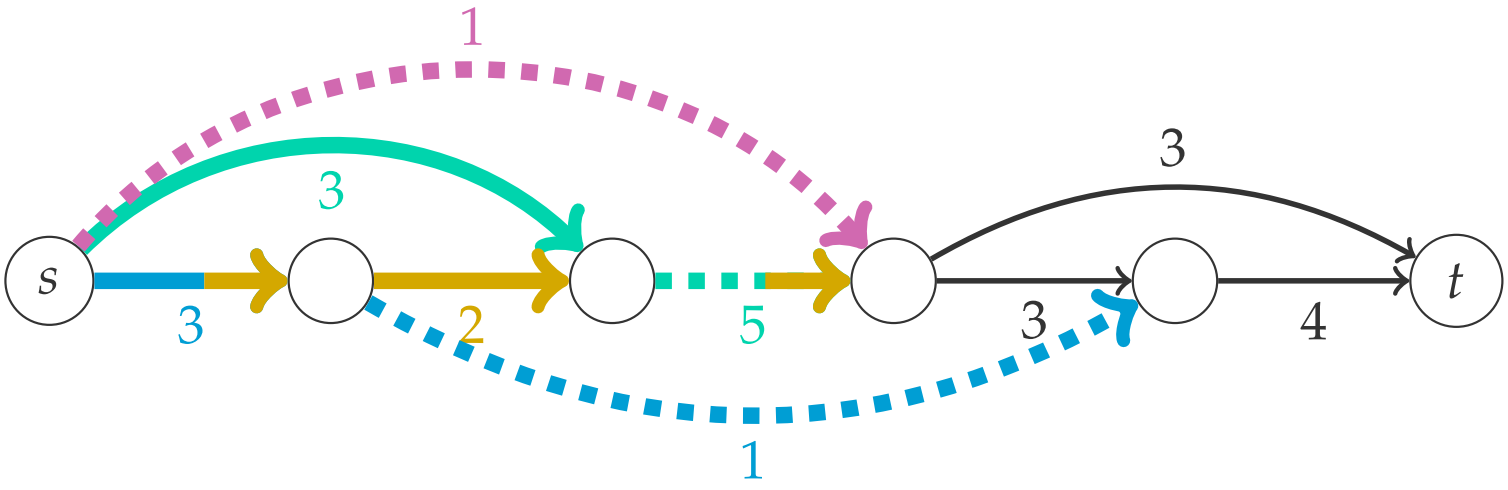


$$w_{\blacksquare} + w_{\blacksquare} = 3$$

$$w_{\blacksquare} = 3$$

$$w_{\blacksquare} = 2$$

The Idea

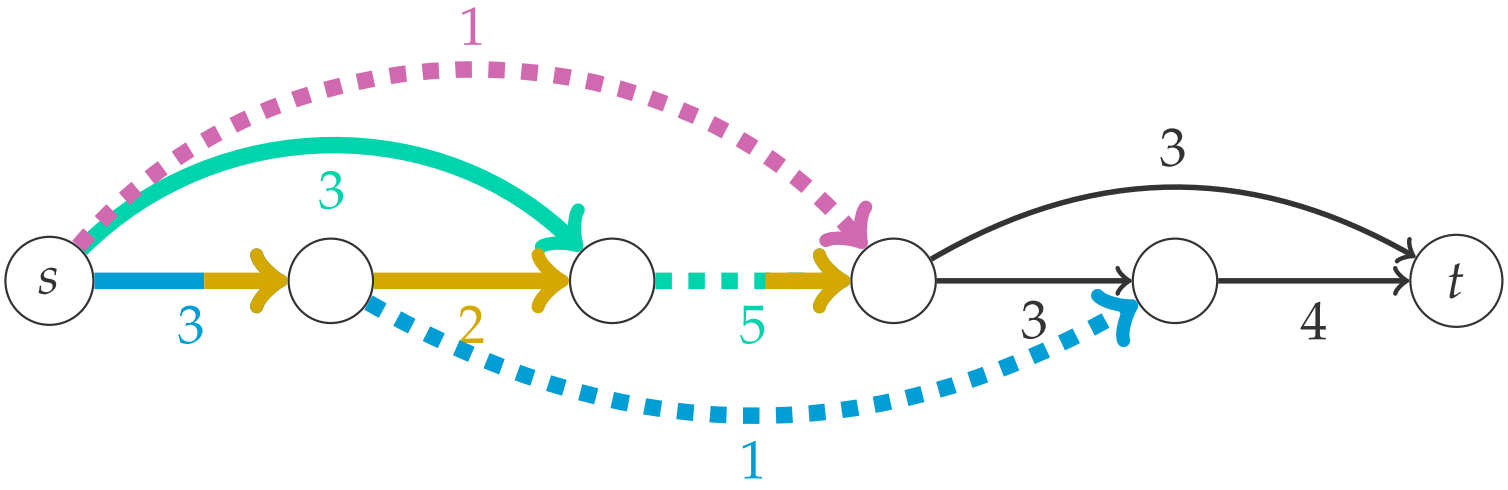


$$w_{\blacksquare} + w_{\blacksquare} = 3 \qquad w_{\blacksquare} = 1$$

$$w_{\blacksquare} = 3 \qquad w_{\blacksquare} + w_{\blacksquare} = 5$$

$$w_{\blacksquare} = 2 \qquad w_{\blacksquare} = 1$$

The Idea

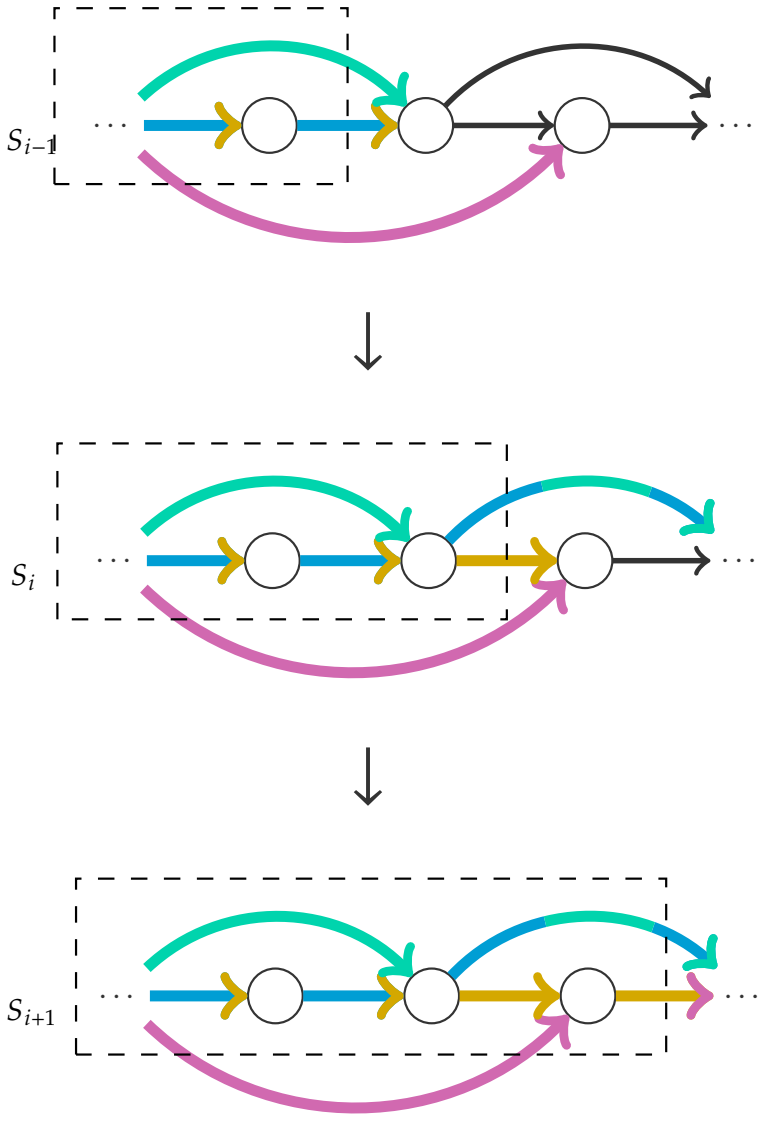


$$w_{\blacksquare} + w_{\blacksquare} = 3$$
$$w_{\blacksquare} = 3$$
$$w_{\blacksquare} = 2$$

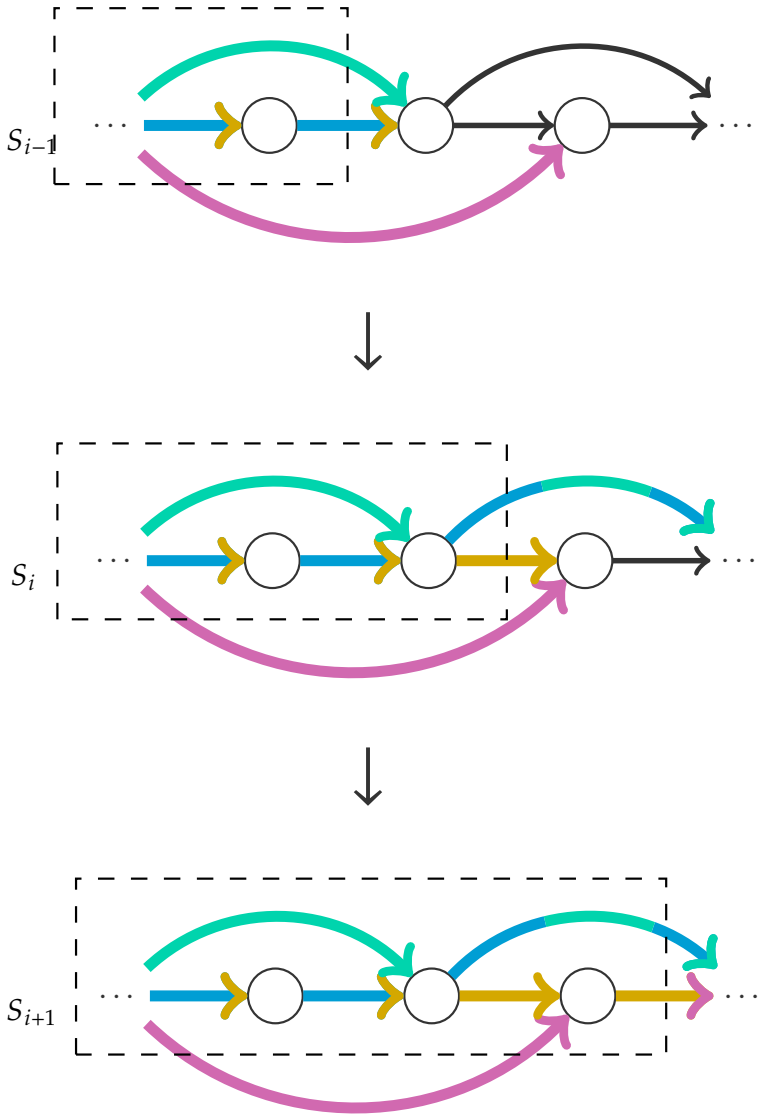
$$w_{\blacksquare} = 1$$
$$w_{\blacksquare} + w_{\blacksquare} = 5$$
$$w_{\blacksquare} = 1$$

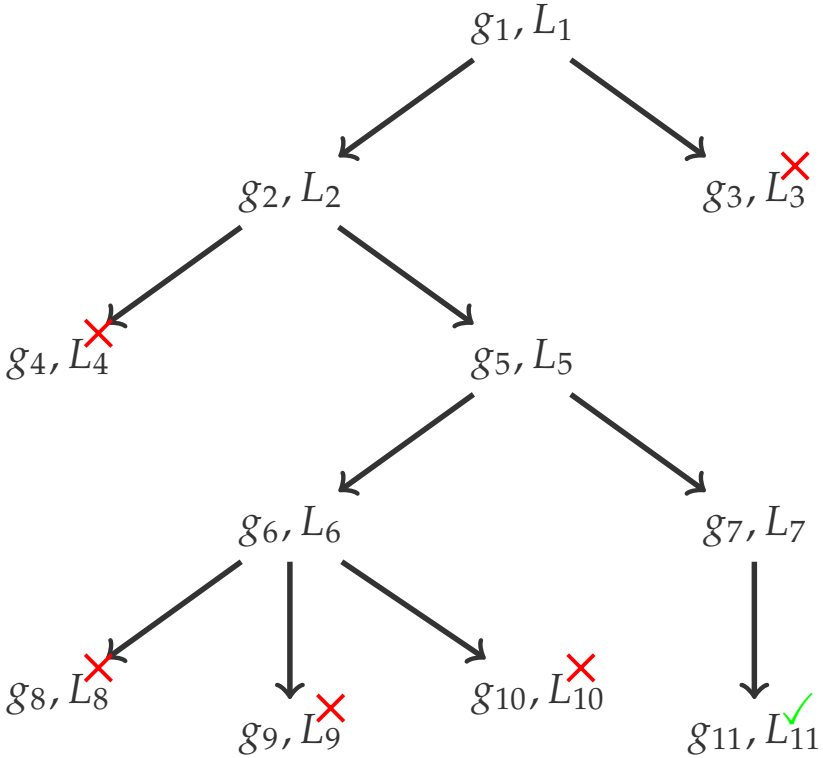
$$Aw = f$$

Dynamic Programming



Dynamic Programming





Conclusion

Conclusion

- Theoretical worst-case runtime linear in n.

Conclusion

- Theoretical worst-case runtime linear in n.
- Competitive runtime with heuristics.

Conclusion

- Theoretical worst-case runtime linear in n.
- Competitive runtime with heuristics.
- Guarantees optimal k.

Conclusion

- Theoretical worst-case runtime linear in n.
- Competitive runtime with heuristics.
- Guarantees optimal k.
- Python already fast, C++ even faster?

Conclusion

- Theoretical worst-case runtime linear in n.
- Competitive runtime with heuristics.
- Guarantees optimal k.
- Python already fast, C++ even faster?

paper: `https://arxiv.org/abs/1706.07851`

github: `https://github.com/theoryinpractice/toboggan`

Kyle Kloster, **Philipp Kuinke**, Michael P. O'Brien, Felix Reidl, Fernando Sánchez Villaamil, **Blair D. Sullivan**, Andrew van der Poel

Thank you!

Supported in part by the Gordon & Betty Moore Foundation's Data-Driven Discovery Initiative through Grant GBMF4560 to Blair D. Sullivan.

RWTH AACHEN UNIVERSITY

NC STATE UNIVERSITY