Transcriptome Assembler

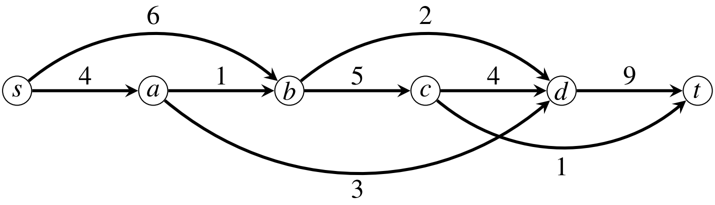
Mingfu Shao

Computational Biology Department, Carnegie Mellon University

January 18, 2016

Problem Statement

- **Input:** strongly connected DAG $G = (V, E)$ with a single source s and a single sink t, weight $w(e)$ for $e \in E$
- **Output:** a set of paths $\mathscr{P}$ from s to t and capacity $c(P)$ for $P \in \mathscr{P}$, such that
  - $|\mathscr{P}|$ is minimized, and that
  - $\sum_{e \in E} |w(e) - \sum_{e \in P} c(P)|$ is minimized



Existing Method: Cufflinks

- **Algorithm:** compute a minimum number of paths to cover all edges using Dilworth's Theorem
- **Disadvantage:** do not consider the weights of edges

Existing Method: Scripture

- **Algorithm:** output all possible paths
- **Disadvantage:** exponential number of paths

Existing Method: IsoLasso

- **Algorithm:** use quadratic programming

$$\begin{array}{rl} \min & \sum_{e\in E} |w(e) - \sum_{e\in P} c(P)|^2 \\ \text{s.t.} & \sum_{P\in\mathscr{P}} c(P) \leq \lambda \end{array}$$

- **Disadvantage:**
  - $|\mathscr{P}|$ is not bounded
  - exponential number of variables

Existing Method: Traph

- **Algorithm:** use a network flow formulation to compute a new weight $w'(e)$ for $e \in E$ such that $\sum_{e \in E} |w(e) - w'(e)|$ is minimized and that there *exists* a set of paths satisfying $\sum_{e \in E} |w'(e) - \sum_{e \in P} c(P)| = 0$ (i.e., there exists a flow decompositions of the new DAG)

- **Disadvantage:**
  - only the weights are updated; the paths are not returned; actually they use a greedy algorithm to compute paths (the same as StringTie)
  - the number of paths is not considerred in this formulation

Existing Method: CLIIQ

- **Algorithm:** use ILP to model this problem
- **Disadvantage:**
  - ILP itself is NP-complete
  - exponential number of variables

Existing Method: StringTie

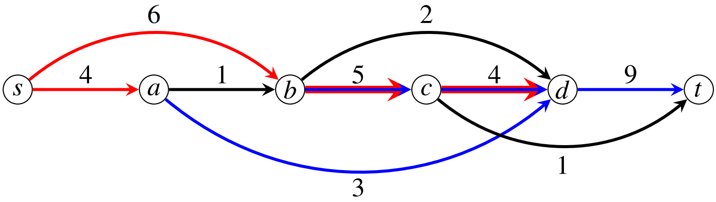
- **Algorithm:**
  - use greedy algorithm to compute paths (transcripts): iteratively compute heaviest path
  - use network flow formulation to estimate abundance (a sophisticated way to handle reads spanning several vertices)
- **Disadvantage:**
  - when compute the current path, the tradeoff between weights is not considerred

Our Algorithm

1. compute a (good) basis $\mathscr{B}$ (with $|E| - |V| + 2$ paths) of the path space
2. use an LP to estimate the capacities of the paths in $\mathscr{B}$
3. try to reduce the number of paths in $\mathscr{B}$
   - for two paths with (almost) identical capacities, merge them into one path
   - discard paths with very small capacities
4. iterate between step 2 and step 3

Compute Basis of the Path Space

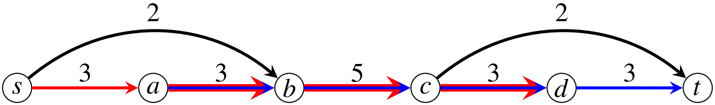
1. for each vertex (except s and t), arbitrarily choose exactly an in-edge and an out-edge (the one with maximum weight);



2. for each vertex v (except s and t), there exists a unique path from s to v, and a unique path from v to t

3. output paths to cover all edges following these chosen edges

Merge Paths: Example



- **Optimal solution:**
  - $P_1^* : s \to a \to b \to c \to d \to t$, capacity $= 3$
  - $P_2^* : s \to b \to c \to t$, capacity $= 2$
- **Our solution:**
  - $P_1 : s \to a \to b \to c \to d \to t$, capacity $= 1$
  - $P_2 : s \to b \to c \to d \to t$, capacity $= 2$
  - $P_3 : s \to a \to b \to c \to t$, capacity $= 2$
- Merge P_2 and P_3
  - $P_4 : s \to b \to c \to t$, capacity $= 2$
  - $P_2 + P_3 = P_4 + P_1$, and P_1 must be in $\mathscr{B}$