Approximation Algorithms for Coflow Scheduling

Erez Kantor   Hamid Jahanjou

Rajmohan Rajaraman

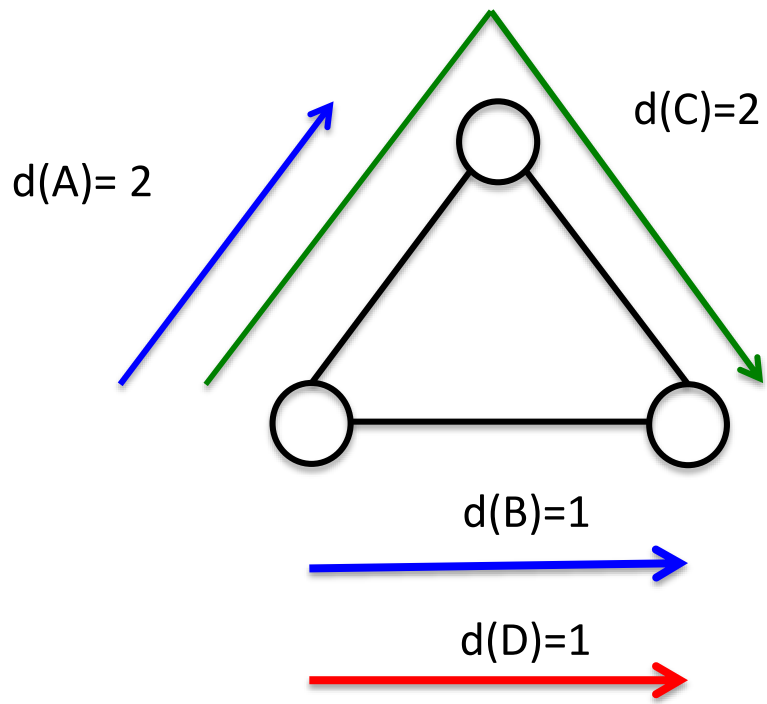
Northeastern University, Boston

Coflows

- Large-scale data processing computations (e.g. MapReduce, Spark, Dryad)
  - Composed of multiple data flows
  - Flows over a shared set of distributed resources
  - Computation completes when all of its flows complete
- Coflow:
  - Collection of flows sharing same performance goal

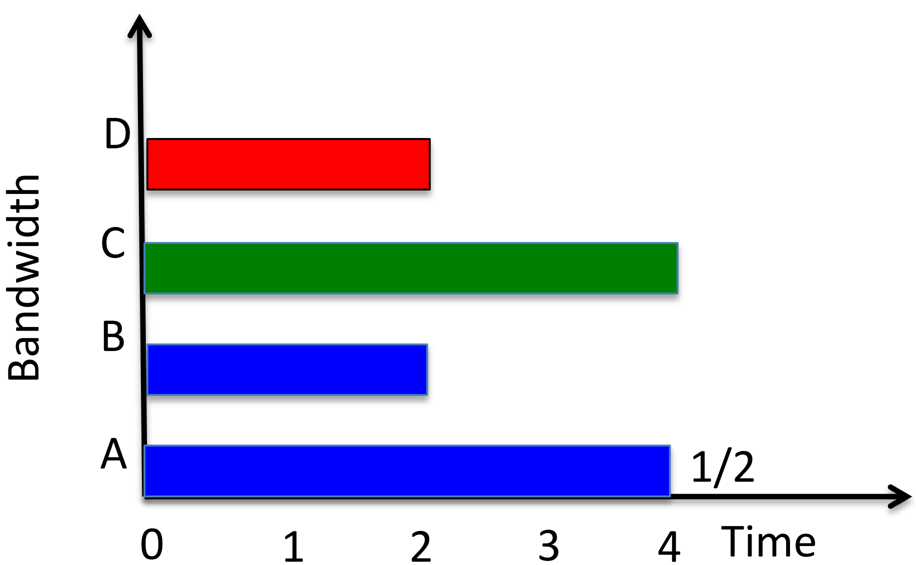
Coflows: An Example

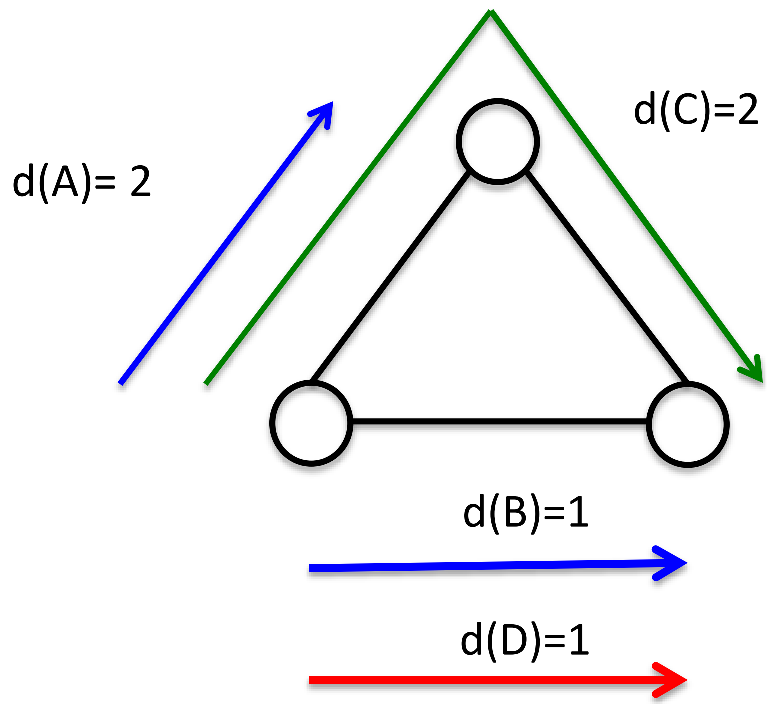
- Blue coflow has two flows
- Red and green coflows have one flow each
- All edge capacities are unit



Coflows: Schedules

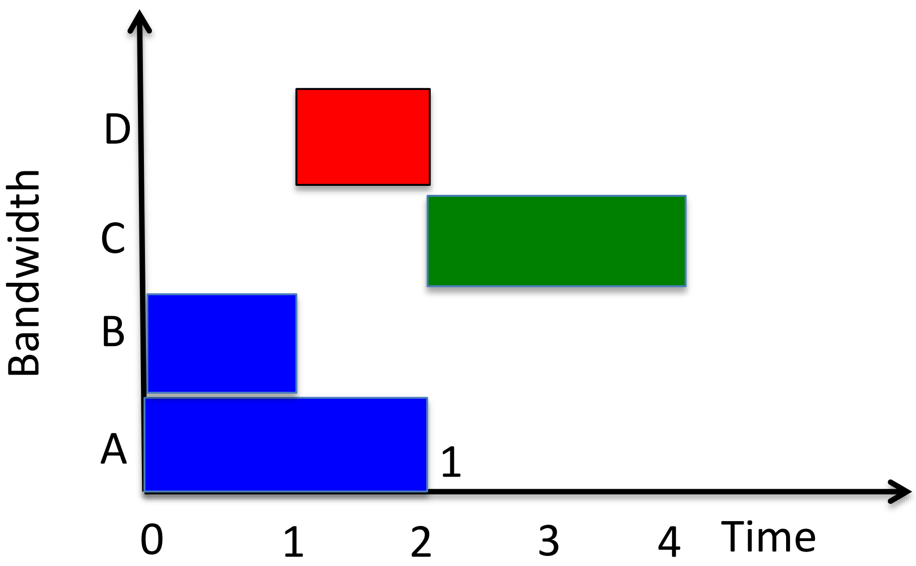
- Schedule 1
  - Constant bandwidth of ½ for all flows
  - 4 + 4 + 2 = 10

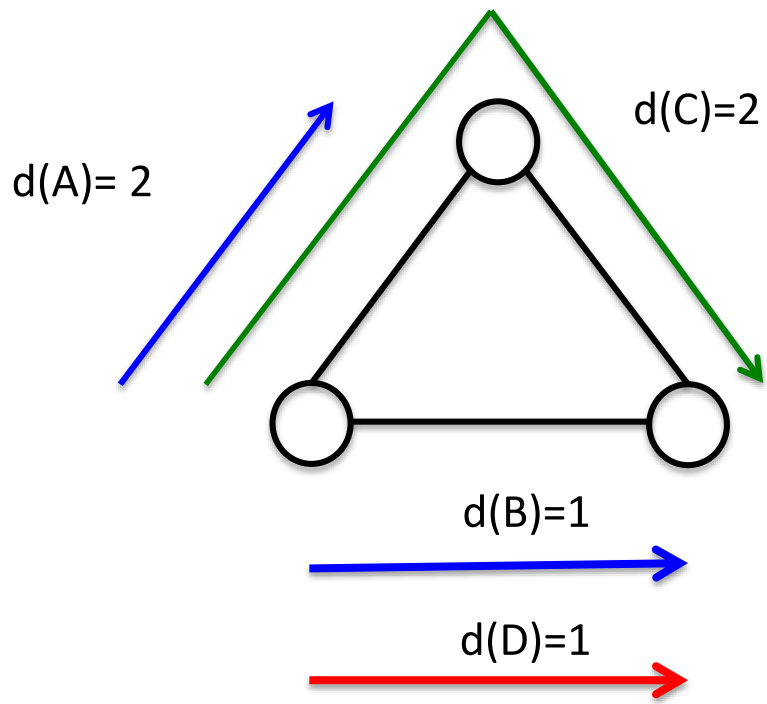




Coflows: Schedules

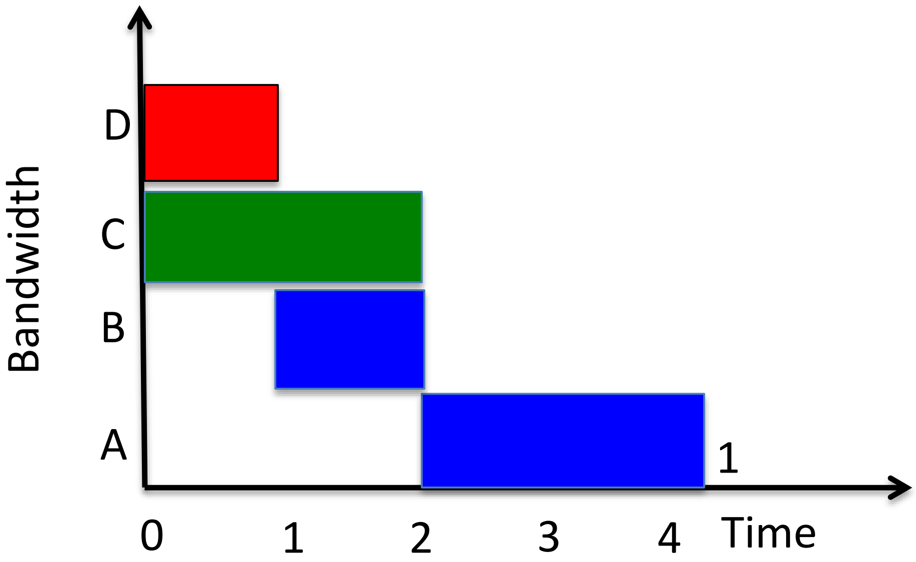
- Schedule 2
  - Blue > Red > Green
  - 2 + 4 + 2 = 8

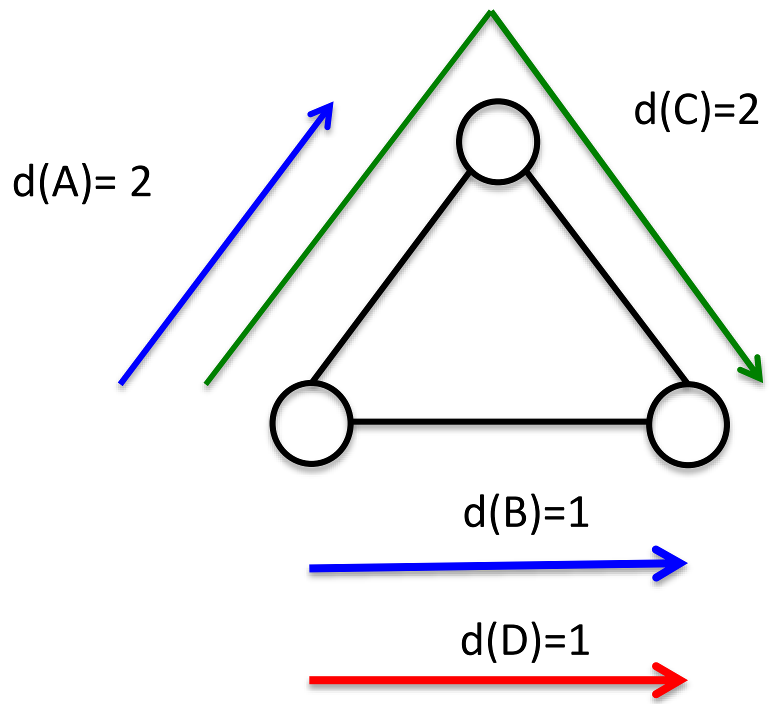




Coflows: Schedules

- Schedule 3
  - Red > Green > Blue
  - 1 + 2 + 4 = 7





Flow Models

Circuits	Bandwidth	Assign paths and bandwidth to source-destination connection requests
Packets	Latency	Route and schedule packets between specified sources and destinations
Tasks	Computation	Schedule tasks on unrelated machines

- In each model, the individual flows share a common objective
  - Completion time: time at which last flow completes

Previous Work

- [Chowdhury-Stoica 2012] introduce coflows as an abstraction for cluster applications
- [Zhao et al 2015] present RAPIER
  - Heuristics for joint scheduling and routing
  - Explicit routing using SDN and bandwidth enforcement using Linux Traffic Control
- [Qui-Stein-Zhong 2015] present constant-factor approximations for coflow scheduling on a non-blocking switch
- More work on scheduling/routing in datacenter networks

New Approximation Algorithms

- **Circuit-based coflows**
  - 4-approximation when paths are given
  - O(log(n)/loglog(n)) approx. when paths not given
- **Packet-based coflows**
  - Constant-approximation in both cases
- **Task-based coflows**
  - Constant-approximation
- Asymptotically optimal modulo standard complexity assumptions [Garg-Kumar-Pandit 2007,Chuzhoy-Guruswami-Khanna-Talwar 20]

Circuit-Based Coflow Scheduling

- Network with edge capacities
- Connection requests with individual demand, source-destination pair, and release time
- Requests are grouped into coflows; each coflow has a weight
- Determine paths and bandwidth assignment over time for each request to minimize weighted average completion time

Circuit-Based Coflows

- Flow i :
  - Source $s(i)$, destination $t(i)$
  - Demand $d(i)$, release $r(i)$
- Coflow j: Set of flows
- Network $G=(V,E)$
- Capacity $c(e)$ for edge e
- Output:
  - For each flow i and time t : $b(i,e,t)$
- Constraints:
  - $b()$ forms a flow for each t

$$\boxed{\int_t \left(\sum_{e \text{ out of } s(i)} b(i,e,t)dt \right) \geq d(i)}$$

  - For each t,

$$\boxed{\sum_{e \text{ out of } s(i)} b(i,e,t) \leq c(e)}$$

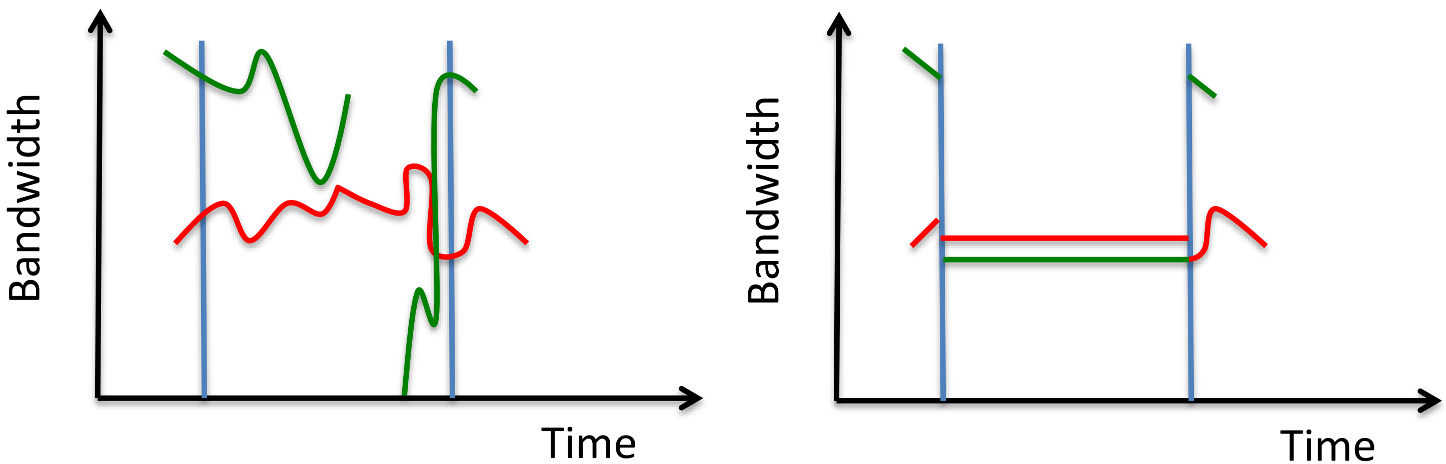
- Objective:

$C(i) =$ completion time of i
$C(j) = \max C(i)$ over flow i in j

$$\boxed{\min \sum_j w(j)C(j)}$$

Piecewise Constant Bandwidth

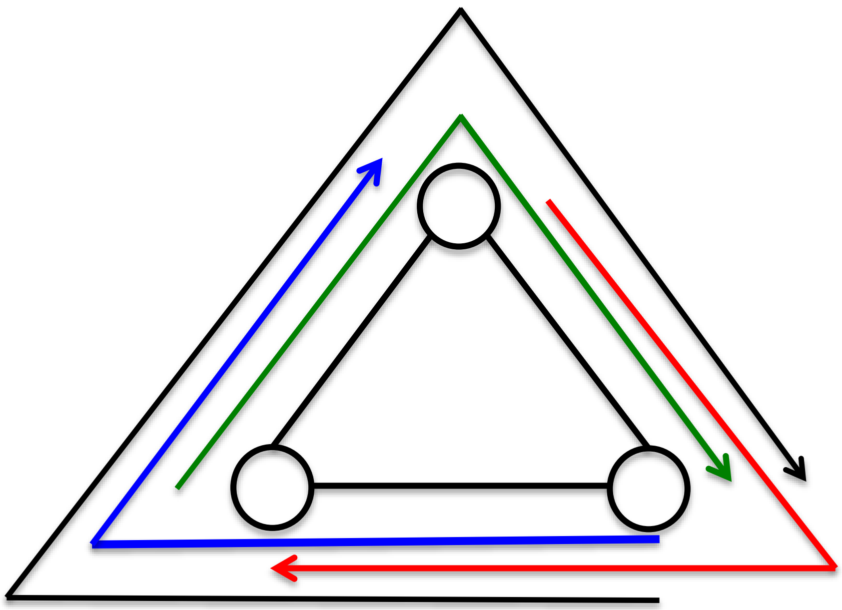
- Lemma: There exists an optimal solution in which between any two events, the bandwidth for any given flow is constant across time.



- Assign average bandwidth over the interval
- Since capacity constraint satisfied at every instant, the new assignment also satisfied

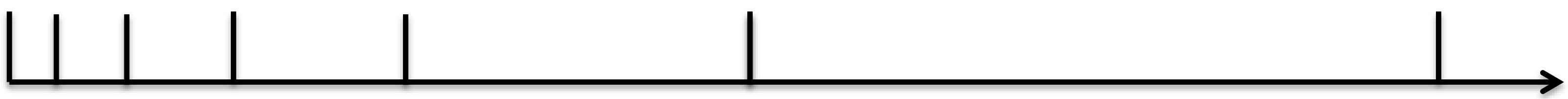
Is There an Optimum Priority Order?

- Optimal schedule:
  - Assign ½ to blue, red, and green for 2 units
  - Assign 1 to black at time 3
  - 2 + 2 + 2 + 3 = 9
- No two flows can be fully scheduled in parallel
  - Every priority order yields 1 + 2 + 3 + 4 = 10



Interval-indexed Linear Program

- Piecewise constant bandwidth allows us to develop a linear program relaxation that achieves a 2-approximation
- Divide time into [0,1), [1,2), ..., $[2^{k-1},2^k)$, ...



- LP(k) for interval k:
  - Constant bandwidth $b_k(i)$ for flow i
  - Edge capacity constraints
- Cross-interval constraint: $\boxed{\sum_k 2^{k-1} b_k(i) \geq d(i)}$
- Objective: $\boxed{\min \sum_j \left(w(j) \max_{\text{flow } i \text{ in } j} \left(2^{2k-1} b_k(i) \right) \right)}$

Interval-Indexed Linear Program

$$\text{Minimize} \sum_i \sum_j \omega'_{ij} c^i_j \qquad \textit{subject to}$$

$$\sum_{\ell} x^i_{j\ell} = 1 \qquad \forall i, j \qquad (31)$$

$$\sum_{\ell \leq L} \tau_\ell x^i_{j\ell} \leq c^i_j \leq \sum_{\ell \leq L} \tau_{\ell+1} x^i_{j\ell} \qquad \forall i, j \qquad (32)$$

$$c^i_j \leq c^i_0 \qquad \forall i, j \qquad (33)$$

$$b^i_{j\ell} = \sigma^i_j x^i_{j\ell} / \tau_\ell \qquad \forall i, j, \ell \qquad (34)$$

$$\sum_{f^i_j \in P(e)} b^i_{j\ell} \leq c(e) \qquad \forall \ell, e \qquad (35)$$

$$r^i_j > \tau_{\ell+1} \Rightarrow x^i_{j\ell} = 0 \qquad \forall i, \ell \qquad (36)$$

$$x^i_{j\ell} \geq 0 \qquad \forall i, j, \ell, e \qquad (37)$$

Constant-Factor Approximation

- Solve the interval-indexed LP
- Assign each flow to the interval following the first one by which ½ of flow completes
- In each interval:
  - Allocate constant bandwidth to each flow assigned so that its demand completes
  - LP constraints and the interval structure guarantee capacity constraints

- High-level takeaway:
  - Can group coflows into priority groups (intervals)
  - Within each group, coflows bandwidth shares are well-specified

When Paths are not Given

- Solve the interval-indexed linear program
- Assign flows to intervals as before
- For each flow:
  - Use the LP bandwidth assignment to decompose into path bandwidth assignments
  - Apply randomized rounding [Raghavan-Thompson 1987] to select a single path for each flow
  - Stretch time by O(log(n)/loglog(n))-factor to achieve desired approximation while satisfying constraints

Packet-Based Coflows

- Network with edge capacities
- Packet requests with individual demand, source-destination pair, and release time
- Requests grouped into coflows with weights
- Determine routing schedule for each packet so as to minimize weighted average completion time
- Key differences from circuit-based model:
  - Models latency and store-and-forward routing
  - Notion of packets as indivisible entities

Packet-Based Coflows

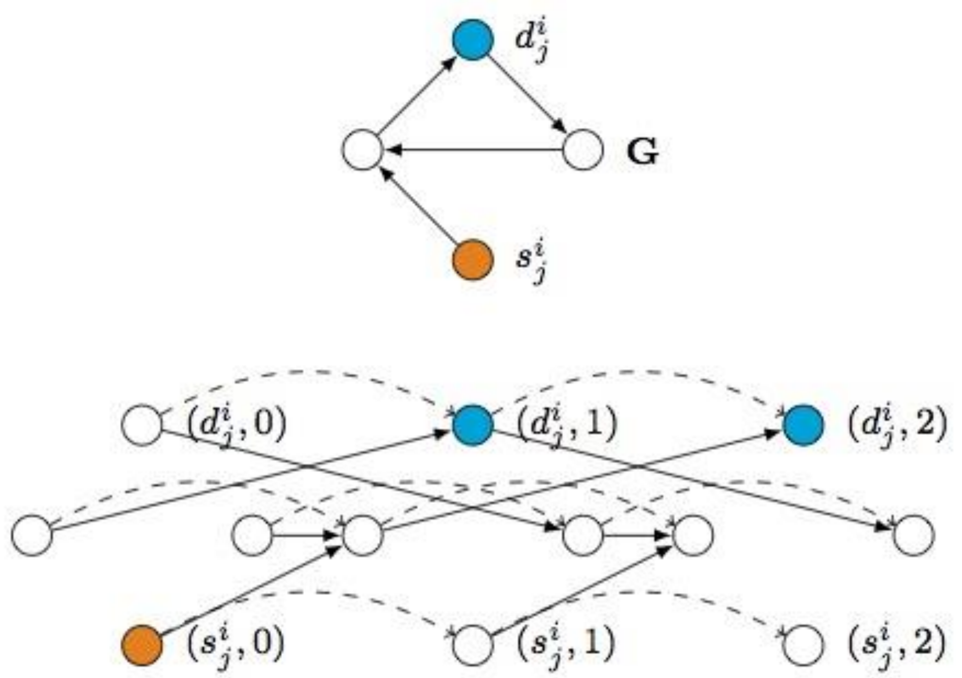


Figure 1: An example graph G (above) and its time-expanded version G^T for $T = 2$ (below). Packet f_j^i needs to be routed from node s_j^i to node d_j^i in G. Corresponding to f_j^i, flows of combined size 1 are sent from $(s_j^i, 0)$ to (d_j^i, k) for $k = 1, 2$. Dashed lines correspond to queue edges.

Algorithm for Packet-Based Coflows

- Ingredients:
  - Interval-index linear program
  - [Leighton-Maggs-Rao 1994] existence of schedule
  - [Leighton-Maggs-Richa-Rao] and more recent work on Lovasz Local Lemma for constructing schedules
  - [Srinivasan-Teo 2001] for finding paths
- Constant-factor approximation

Future Directions

- Evaluation of algorithms in practice
  - Can we avoid solving the interval-indexed LPs?
  - In certain cases involving special topologies like paths and trees:
    - Can get simpler and better algorithms using total unimodularity
  - Improve the hidden constants in approx ratio
  - Improve bounds for restricted classes of coflows
    - E.g., flows in a coflow share a common source

Future Directions

- Other objective functions
  - Minimize average weighted response time
  - Cost-based objectives
- Other models
  - Wavelength allocation in optical networks
    - Strong hardness of approximation
    - For paths, interesting connections to the well-studied Unsplittable Flow Problem
- Online scheduling of coflows