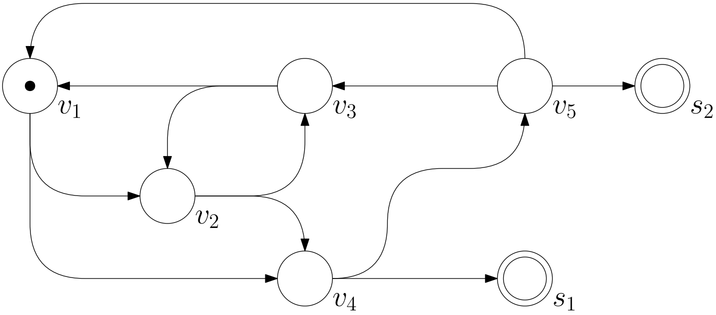
A Quasi-Polynomial Time Algorithm for Multi-Arrival on Tree-Like Multigraphs

Ebrahim Ghorbani, Jonah Leander Hoff, Matthias Mnich

STACS 2025

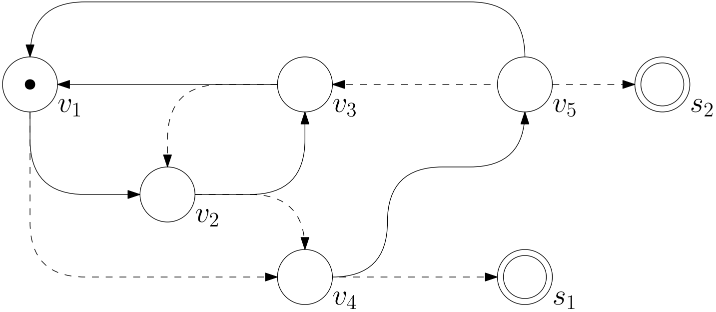
Arrival

- Arrival: which sink does the particle end up on?
- Each vertex: routes along outgoing arcs in fixed cyclic order

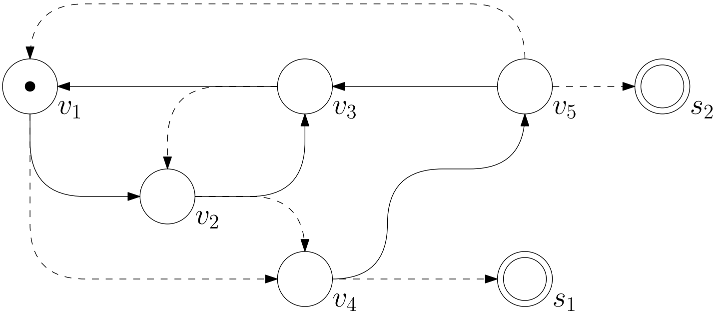


Arrival

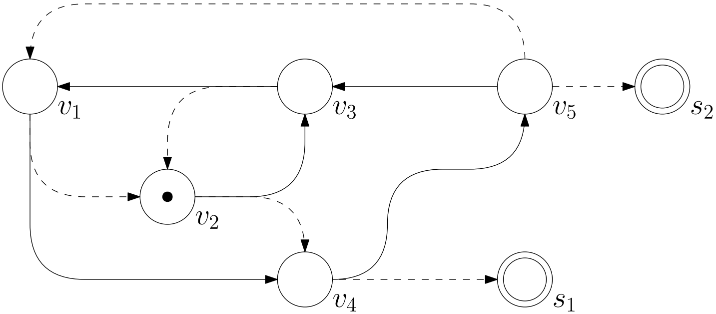
- Arrival: which sink does the particle end up on?
- Each vertex: routes along outgoing arcs in fixed cyclic order



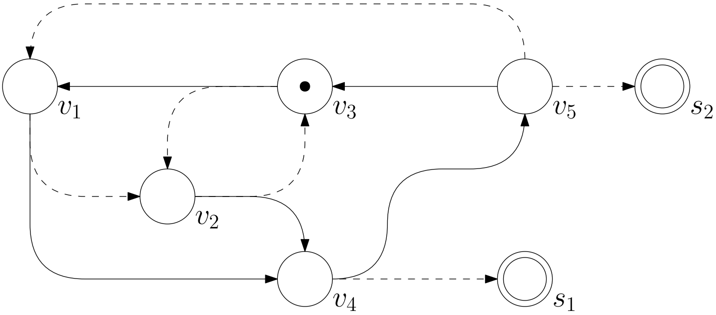
Routing Single Particle



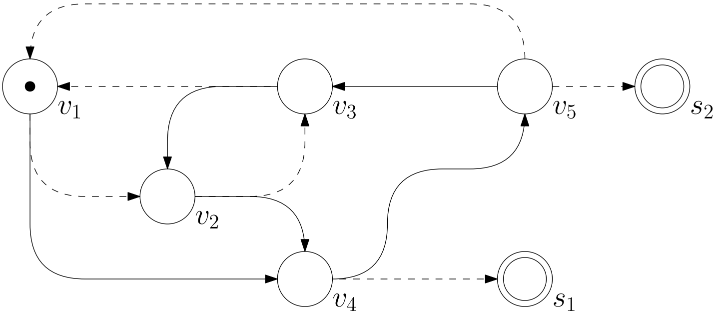
Routing Single Particle



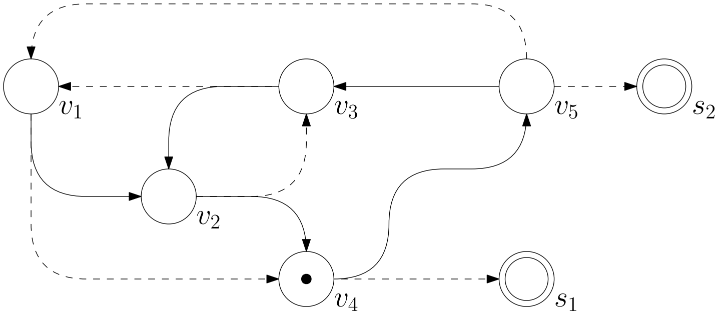
Routing Single Particle



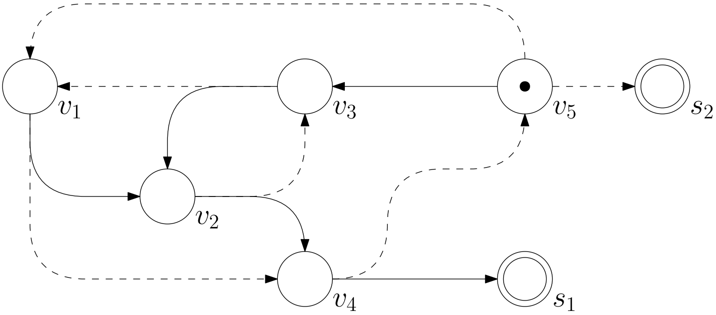
Routing Single Particle



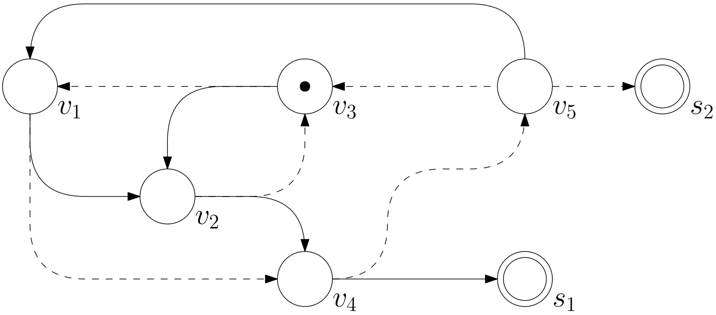
Routing Single Particle



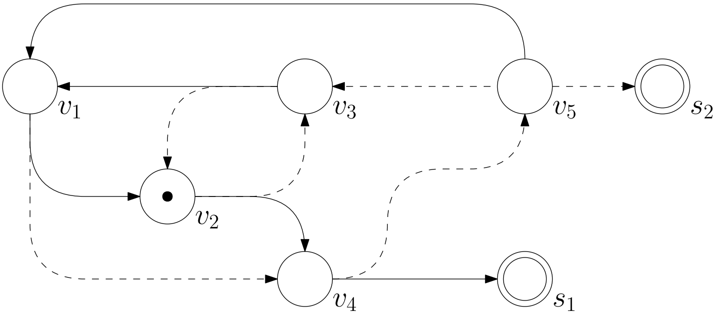
Routing Single Particle



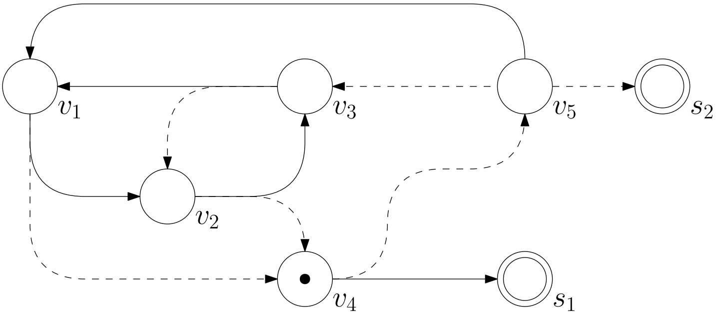
Routing Single Particle



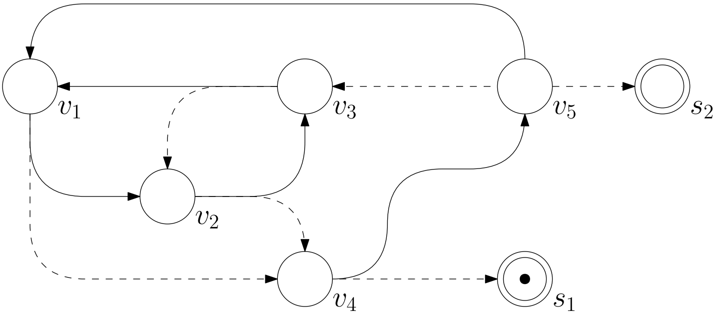
Routing Single Particle



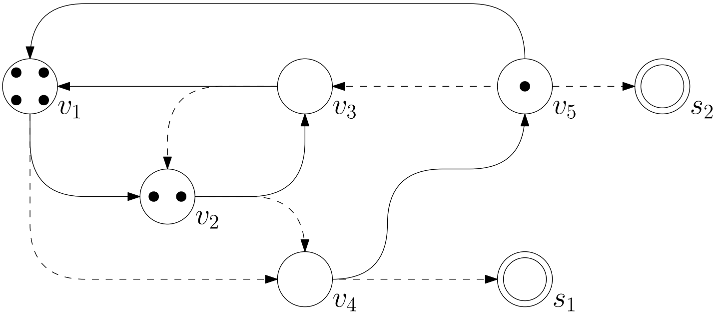
Routing Single Particle



Routing Single Particle

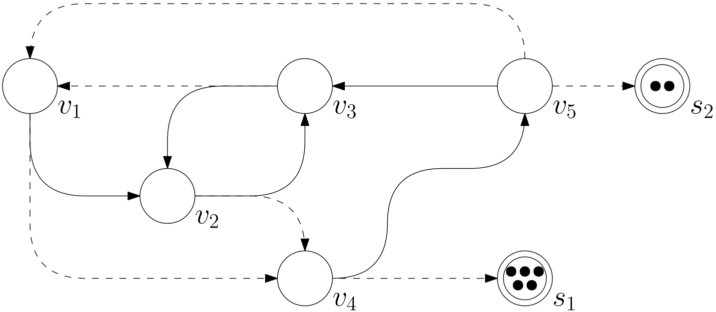


Routing Multiple Particles



- Any maximal routing sequence converges to the same final configuration
- Multi-Arrival: how many particles end up on each sink?

Routing Multiple Particles



- Any maximal routing sequence converges to the same final configuration
- Multi-Arrival: how many particles end up on each sink?

Result

- Multi-Arrival on tree-like multigraphs in time $\mathcal{O}^*(\log^\kappa m)$
- $\kappa \approx$ measure of balanced nested branchings
  - Path-Like: $\kappa = 1$
  - Tree-Like: $\kappa \leq \log n_b$, where n_b number of branching vertices

Prior Work

Type	Complexity	Graph	Reference
Arrival	$2^{\mathcal{O}(\sqrt{n}\log n)}$	Any	[1]
Arrival	$\mathcal{O}^*(1)$	Tree-like	[2]
Multi-Arrival	$\mathcal{O}^*(1)$	Uniform path-like	[3]
Multi-Arrival	$\mathcal{O}^*(1)$	Path-like	This paper
Multi-Arrival	$\mathcal{O}^*(\log^{\kappa} m)$	Tree-like	This paper

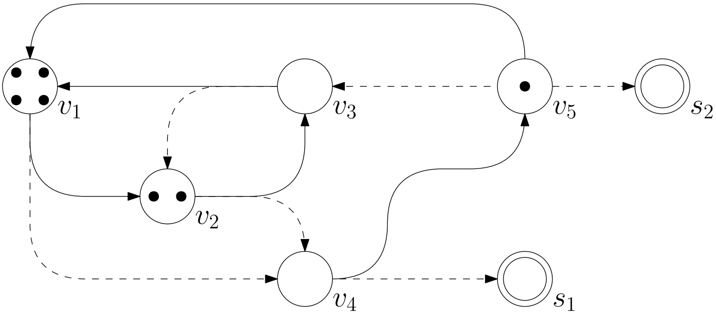
- If Multi-Arrival $\mathcal{O}^*(1)$ on $G[V \setminus X]$ with $|X|$ bounded
  - Then Arrival $\mathcal{O}^*(1)$ on $G[V]$ **[Gärtner, Haslebacher, and Hoang 2021]**

[1] **[Gärtner, Haslebacher, and Hoang 2021]**
[2] **[Auger, Coucheney, and Duhaze 2022]**
[3] **[Auger, Coucheney, Duhazé, et al. 2023]**

Table of Contents

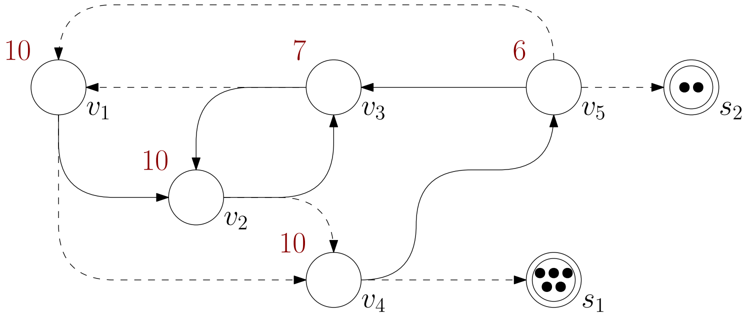
Legal Routing Vectors

- Routing (vector) $r \in \mathbb{Z}^V$: how often each vertex is routed
- Legal routing: corresponds to some legal routing sequence
- Maximal legal routing: legal routing with all particles on sinks



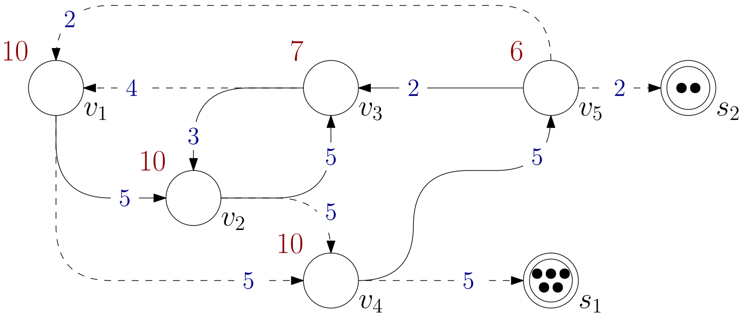
Legal Routing Vectors

- Routing (vector) $r \in \mathbb{Z}^V$: how often each vertex is routed
- Legal routing: corresponds to some legal routing sequence
- Maximal legal routing: legal routing with all particles on sinks



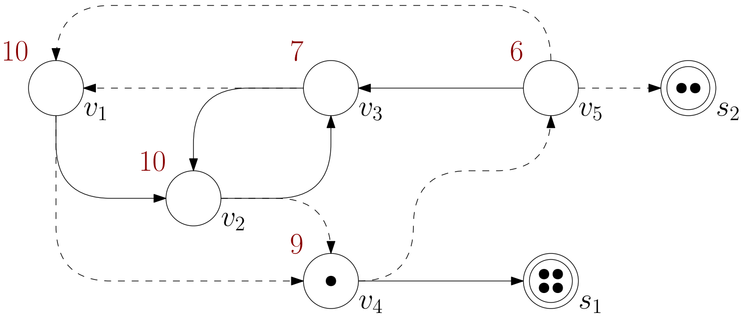
Compensated Routing Vectors

- Compensated routing: each vertex has out-flow less or equal in-flow
- Lower-bound flow into sinks
- Switching flows **[Dohrau et al. 2017]**



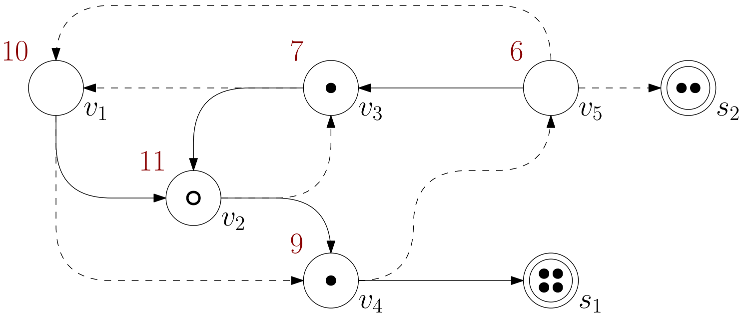
Compensated Routing Vectors

- Compensated routing: each vertex has out-flow less or equal in-flow
- Lower-bound flow into sinks
- Switching flows **[Dohrau et al. 2017]**



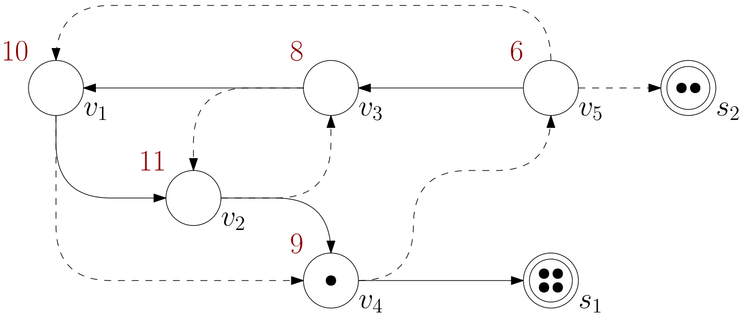
Compensated Routing Vectors

- Compensated routing: each vertex has out-flow less or equal in-flow
- Lower-bound flow into sinks
- Switching flows **[Dohrau et al. 2017]**



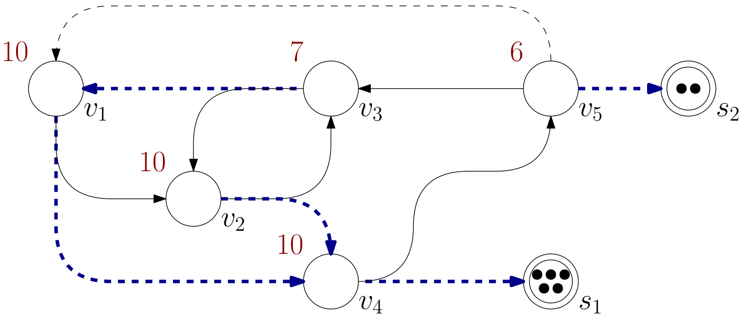
Compensated Routing Vectors

- Compensated routing: each vertex has out-flow less or equal in-flow
- Lower-bound flow into sinks
- Switching flows **[Dohrau et al. 2017]**



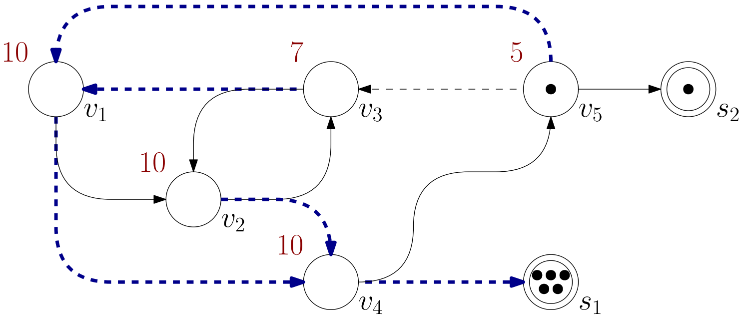
Directed Routing Vectors

- Idea: maximize flow into fixed sink *s*
- *s*-directed routing: last particles sent were towards *s*



Directed Routing Vectors

- Idea: maximize flow into fixed sink s
- s-directed routing: last particles sent were towards s



Problem

Original problem:

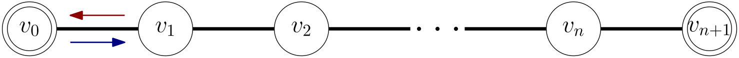
$$\begin{array}{ll} \text{maximize} & \text{flow of } \hat{r} \text{ into sinks} \\ \text{subject to} & \hat{r} \in \mathbb{Z}^V \text{ is compensated.} \end{array}$$

Directed problem for sink s:

$$\begin{array}{ll} \text{maximize} & \text{flow of } r_s \text{ into } s \\ \text{subject to} & r_s \in \mathbb{Z}^V \text{ is } s\text{-directed and compensated.} \end{array}$$

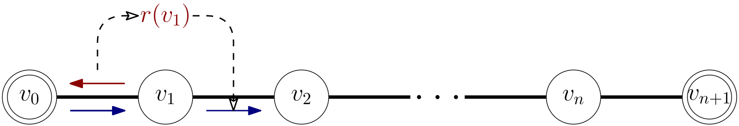
Path-Like Case

1. Guess flow $v_0 \leftarrow v_1$
2. Iteratively construct r
   - Since s-directed: flow $v_{i-1} \leftarrow v_i$ gives $r(v_i)$
   - Since compensated: $r(v_i) \leq \sigma_i + (v_{i-1} \rightarrow v_i) + (v_i \leftarrow v_{i+1})$
3. If missing flow $v_n \leftarrow v_{n+1}$ is zero: guess is achievable
4. Otherwise: guess is too large



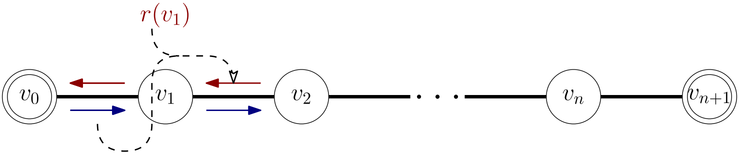
Path-Like Case

1. Guess flow $v_0 \leftarrow v_1$
2. Iteratively construct r
   - Since s-directed: flow $v_{i-1} \leftarrow v_i$ gives $r(v_i)$
   - Since compensated: $r(v_i) \leq \sigma_i + (v_{i-1} \rightarrow v_i) + (v_i \leftarrow v_{i+1})$
3. If missing flow $v_n \leftarrow v_{n+1}$ is zero: guess is achievable
4. Otherwise: guess is too large



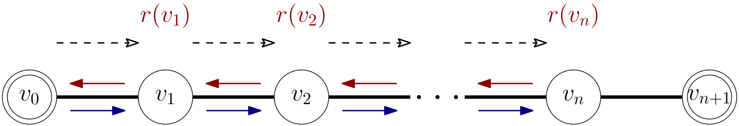
Path-Like Case

1. Guess flow $v_0 \leftarrow v_1$
2. Iteratively construct r
   - Since s-directed: flow $v_{i-1} \leftarrow v_i$ gives $r(v_i)$
   - Since compensated: $r(v_i) \leq \sigma_i + (v_{i-1} \rightarrow v_i) + (v_i \leftarrow v_{i+1})$
3. If missing flow $v_n \leftarrow v_{n+1}$ is zero: guess is achievable
4. Otherwise: guess is too large



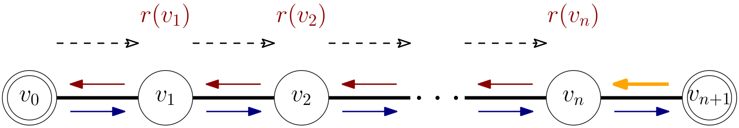
Path-Like Case

1. Guess flow $v_0 \leftarrow v_1$
2. Iteratively construct r
   - Since s-directed: flow $v_{i-1} \leftarrow v_i$ gives $r(v_i)$
   - Since compensated: $r(v_i) \leq \sigma_i + (v_{i-1} \rightarrow v_i) + (v_i \leftarrow v_{i+1})$
3. If missing flow $v_n \leftarrow v_{n+1}$ is zero: guess is achievable
4. Otherwise: guess is too large



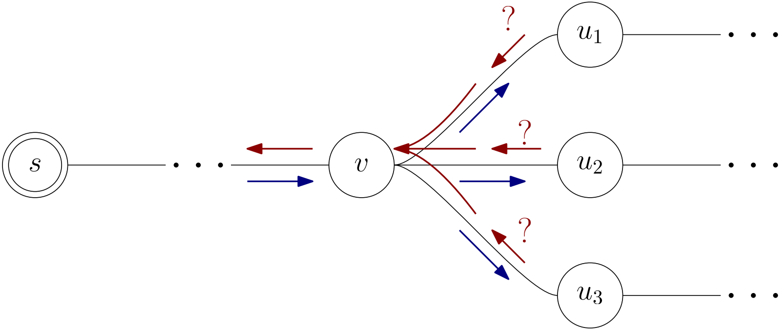
Path-Like Case

1. Guess flow $v_0 \leftarrow v_1$
2. Iteratively construct r
   - Since s-directed: flow $v_{i-1} \leftarrow v_i$ gives $r(v_i)$
   - Since compensated: $r(v_i) \leq \sigma_i + (v_{i-1} \rightarrow v_i) + (v_i \leftarrow v_{i+1})$
3. If missing flow $v_n \leftarrow v_{n+1}$ is zero: guess is achievable
4. Otherwise: guess is too large



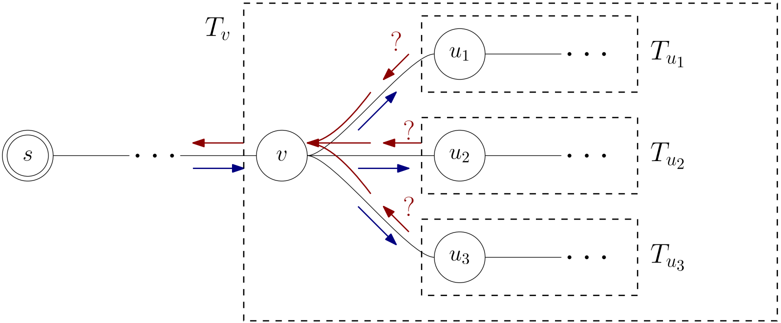
Tree-Like Extension

- Can only infer $v \leftarrow u_1 + u_2 + \ldots$
- Each $v \leftarrow u_i$ upper-bound depends on $v \rightarrow u_i$
- Solve by recursion with fixed $v \leftarrow u_j$ inferred



Tree-Like Extension

- Can only infer $v \leftarrow u_1 + u_2 + \dots$
- Each $v \leftarrow u_i$ upper-bound depends on $v \rightarrow u_i$
- Solve by recursion with fixed $v \leftarrow u_j$ inferred



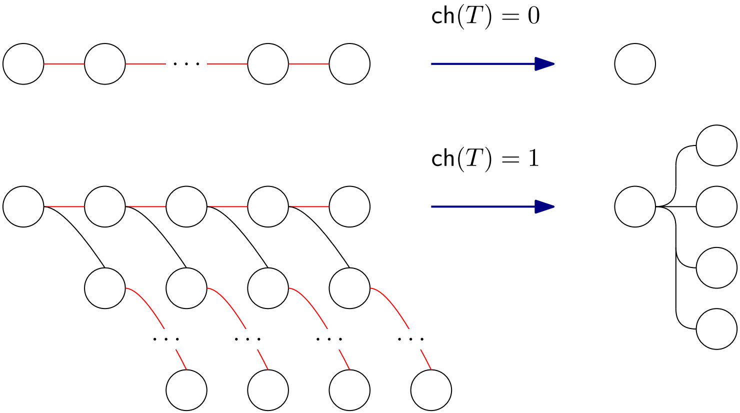
Contracted-Height

- $\kappa = 1 + \max\{\mathrm{ch}(T_r) \mid r \in N^-(S)\}$
- *Contracted height* $\mathrm{ch}(T_r)$: minimal height obtained by contracting per vertex one edge to child

$$\mathrm{ch}(T_v) = \begin{cases} 1 + \mathrm{ch}(T_{u_1}) & \mathrm{ch}(T_{u_1}) = \mathrm{ch}(T_{u_2}) \\ \mathrm{ch}(T_{u_1}) & \mathrm{ch}(T_{u_1}) > \mathrm{ch}(T_{u_2}) \end{cases}$$

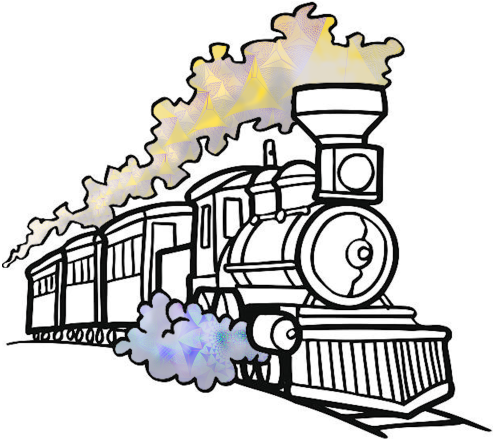
Contracted-Height

$$\mathsf{ch}(T_v) = \begin{cases} 1 + \mathsf{ch}(T_{u_1}) & \mathsf{ch}(T_{u_1}) = \mathsf{ch}(T_{u_2}) \\ \mathsf{ch}(T_{u_1}) & \mathsf{ch}(T_{u_1}) > \mathsf{ch}(T_{u_2}) \end{cases}$$



Summary

- Decomposition of maximal legal routing into s-directed compensated routings
- Recursive search to find s-directed compensated routings
  - Linear approximation guides exponential search
  - Recursion is contracted on one sub-tree for each vertex
- Dynamic program to find s-directed compensated routings



Thank you for your attention!

Bibliography I

Auger, David, Pierre Coucheney, and Loric Duhaze (2022). "Polynomial Time Algorithm for ARRIVAL on Tree-like Multigraphs". In: *arXiv e-prints*, arXiv–2204.

Auger, David, Pierre Coucheney, Loric Duhazé, et al. (2023). "Generalized ARRIVAL Problem for Rotor Walks in Path Multigraphs". In: *Reachability Problems*. Ed. by Olivier Bournez, Enrico Formenti, and Igor Potapov. Vol. 14235. Cham: Springer Nature Switzerland, pp. 183–198. ISBN: 978-3-031-45285-7 978-3-031-45286-4. DOI: 10.1007/978-3-031-45286-4_14. (Visited on 07/07/2024).

Dohrau, Jérôme et al. (2017). "ARRIVAL: A Zero-Player Graph Game in NP ∩ coNP". In: *A Journey Through Discrete Mathematics: A Tribute to Jiří Matoušek*. Ed. by Martin Loebl, Jaroslav Nešetřil, and Robin Thomas. Cham: Springer International Publishing, pp. 367–374. ISBN: 978-3-319-44479-6. DOI: 10.1007/978-3-319-44479-6_14. (Visited on 05/03/2022).

Bibliography II

Gärtner, Bernd, Sebastian Haslebacher, and Hung P. Hoang (2021). “A Subexponential Algorithm for ARRIVAL”. In: *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*. Vol. 198. Schloss Dagstuhl-Leibnitz-Zentrum für Informatik, p. 69.