New Balancing Methods for Parametric Maximum Flow Problems

Bin Zhang/HPL

Prepared for EE380 at

Stanford University

3-14-2007

Acknowledgements

- Julie Ward, Krishna Venkatraman
  - Revenue Coverage Model (INFORMS, Philadelphia 1999)
- Qi (Annabelle) Feng
  - Implemented the maximum flow algorithm for the Revenue Coverage Problem
  - based on *Improved Algorithm for Bipartite Network Flow*, by Ahuja at el. (1994)

- **Other Related Work**
- Robert Tarjan, Yunhong Zhou, Jia Mao
  - Applied balancing method to non-parametric maximum flow problem with a general network topology.
  - Gave sufficient conditions for stopping.
- Maxim Babenko, Jonathan Derryberry, Andrew Goldberg, Robert Tarjan, Yunhong Zhou
  - *"Experimental Evaluation of Parametric Maximum-Flow Algorithms"*, WEA'07
  - Experimental comparison of Star-balancing algorithm and GGT algorithm on bipartite parametric flow networks using both real-world large data sets and synthetic data sets
  - GGT → Gallo, Grigoriadis & Tarjan (1989)

Outline

- Motivation: Product Proliferation Problem
  - Revenue Coverage Optimization
  - integer programming (IP) model
  - Converting IP model to a parametric maximum-flow problem
- Discovery: New balancing methods for parametric maximum flow problem
  - Path-Balancing and Star-Balancing (old name: Vertex-Balancing)
  - Experimental evaluations
- Other related work

Focus of the presentation

Product Proliferation

Revenue Coverage Optimization

Integer Programming Model

Product Proliferation

- Problem
  - Thousands of active configurations of products (SKUs) on the price list
    - Confusing customer experience
    - High product management costs
    - Poor order fulfillment performance
    - SKU → http://en.wikipedia.org/wiki/Stock_Keeping_Unit
- Observations
  - Small percentage of SKUs generate most of the revenue
  - Some low-revenue SKUs appear in high-revenue orders
- Approach
  - Choose a portfolio (a subset) of SKUs to maintain
    - covers as much revenue (or profit) as possible
    - an optimization problem – Revenue Coverage Optimization
  - Revenue from an order is covered if all SKUs involved in the order are in the portfolio

Integer Programming Formulation

Optimization Problem Find the portfolio *P* of size <= *S* that maximizes the total revenue of orders covered

Variables

x_p = *1* if *p* (a SKU) is in *P*, 0 otherwise

y_o = *1* if order *o* is covered by *P*, 0 otherwise

IP(*S*): $\max \sum_{o \in O} R_o y_o$ Maximize the total covered revenue.

$y_o < x_p \; if \;\; p \in o$ An order is covered only if all *p's* involved are in *P*.

$\sum_p x_p < S$ The portfolio size is constrained.

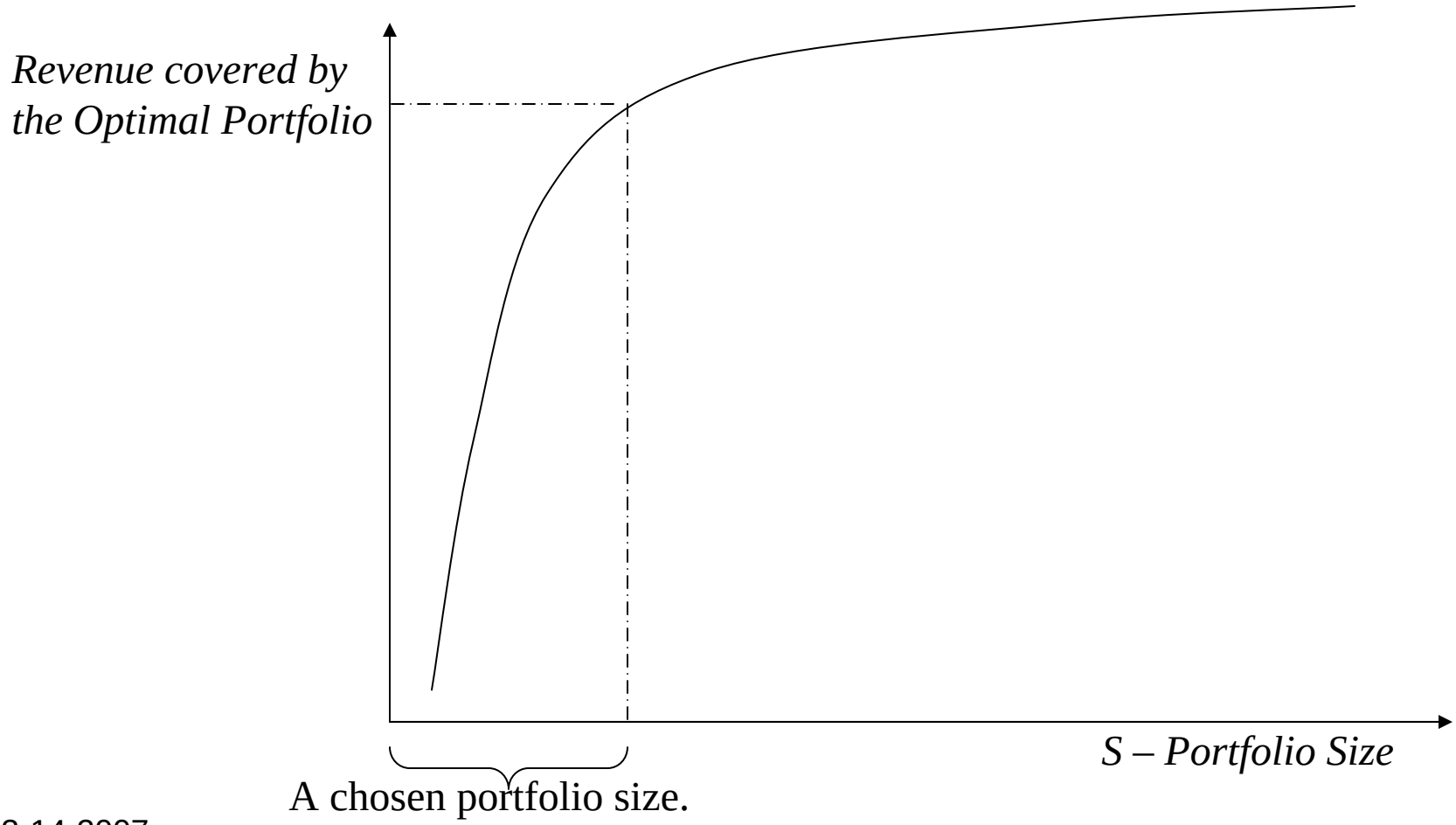
$x_p, y_o \in \{0,1\}$

More on the IP

- The coefficients and dependencies in the IP formulation are from recently shipped orders.
- It is assumed that the near future is similar to the recent past so that the solution to the IP model can be applied to the future.
- IP model is “conservative”:
  - If the portfolio was constrained to *P* in the recent past, some of the orders not covered by *P* could have been different.
  - But we assume that orders not covered by *P* are lost.
- Selected portfolio has an influence on customer’s choice.

Revenue Coverage Curve

Find solutions to IP(S) at as many different values of S as possible to plot a smooth enough revenue coverage curve.



Other Applications

- Also see GGT (1989) paper and D. S. Hochbaum web site:
  - http://www.ieor.berkeley.edu/~hochbaum/
  - Repair kit selection
  - Freight handling terminals problem
  - Database record segmentation
  - ……

Finding Solutions by "Lagrangian relaxation"

IP(S) Formulation:

$$\max \sum_{o \in O} R_o y_o$$

$$y_o < x_p \ \ if \ \ p \in o$$

$$\sum_p x_p < S$$

$$x_p, y_o \in \{0,1\}$$

Linear Programming LP(λ):

$$\max \sum_{o \in O} R_o y_o - \lambda \sum_p x_p$$

$$y_o < x_p \ \ if \ \ p \in o$$

$$0 \le x_p, y_o \le 1$$

Total Unimodularity (Lawler's book on *Combinatorial Optimization*, p160-165) → LP solutions are in integers and are solutions of the original IP

Solutions under different λ are nested.

(Balinski 1970, Rhys 1970)

More Comments on *LP(λ)*

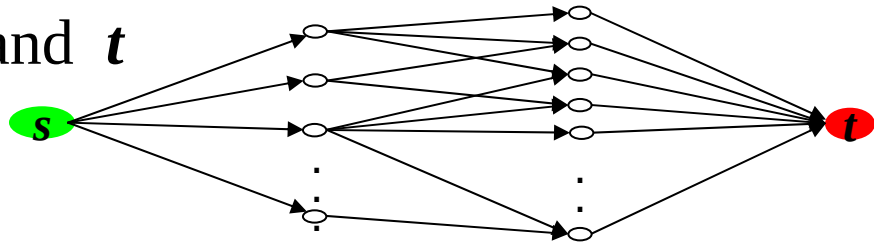
- The LP(λ) under all λ covers only some solutions of IP(*S*):
  - The solutions of the IP problem under different *S* are not necessarily nested.
  - It may miss solutions of IP(*S*) for some *S* even when such solutions are nested with the chain of solutions from LP(λ). Examples can be constructed easily.
- # solutions to LP(λ) is large enough to plot a smooth enough revenue coverage curve because
  - the distribution of the revenue impacted by removing a SKU is highly skewed in the product proliferation problem.
- The LP(λ) problem has millions of constraints: $y_o < x_p \ \ if \ \ p \in o$
  - it took a long time for available LP solvers to find solutions at hundreds of different λ values, which are difficult to choose intelligently before knowing the solutions.
- Much more efficient solvers can be designed after converting the LP(λ) problem into a maximum-flow problem on a capacitated flow network.

A Very Brief Introduction to

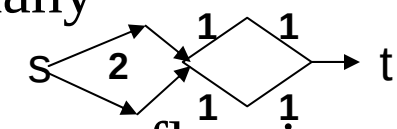
Capacitated Flow-Networks and Maximum-Flow Algorithms

Maximum-flow/Minimum-cut

- A flow network is a directed graph with capacities on its arcs
- with two distinguished vertices: ***s*** and ***t***
- flows at each vertex is balanced
  - rate of flowing in = rate of flowing out



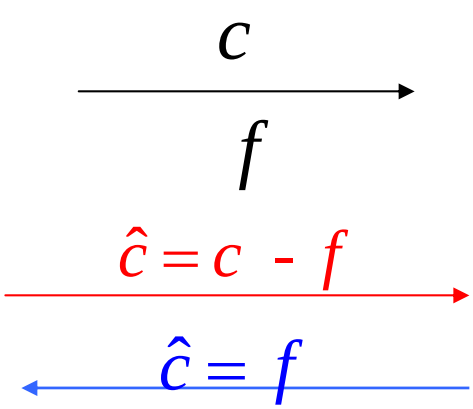
- Maximum-flow problem: what is the maximum capacity of flow that can go from *s* to *t* through the network?
- Minimum-cut problem: where is the bottleneck?
  - A cut is a bipartition of vertices, one includes *s* and the other includes *t*
  - The capacity of a cut is the total capacity of all arcs crossing the partition.
  - Minimum-cut problem: find a cut that has the minimum capacity
- More than one minimum-cut possible and infinitely many realizations of maximum-flows possible



- Ford&Fulkerson (1956) Theorem: the value of a maximum flow is equal to the capacity of the minimum cut.

Residual Capacity, Residual Network and Residual Path

- Working with a capacitated flow network with existing flows
- The maximum amount of flow can be added to an arc is called residual capacity.
- Residual network = adding all reversed arcs to the original network and assign residual capacities to all the arcs.
- A path in the residual network is called a residual path.



See book by Ahuja at el. *Network Flows*

Two Important Max-Flow Algorithms

- Augmenting Flow Algorithm (Ford&Fulkerson 1956)
  - Find an augmenting path from *s* to *t* and add flow to its maximum residual capacity. Keep doing this until no more such paths left.
  - Pros: conceptually very simple
  - Cons: augmenting paths form a huge space, except for simple networks
  - Later improvements (Dinitz 1970)
- Preflow-Push-Relabel Algorithms (Goldberg & Tarjan 1986)
  - The best asymptotic worse-case bound is held by one of these.
  - Turned into an algorithm for parametric flow-networks using recursive binary subdivision of the problem, which finds solutions at all breakpoints of l.
    - GGT: Gallo, Grigoriadis & Tarjan (1989),
  - Implementations:
    - For ordinary maximum flow problems (Cherkassky & Goldberg 1997)
    - For parametric maximum flow problems (Babenco & Goldberg 2006)

Conversion of LP(l) Problem to a Parametric Maximum Flow Problem

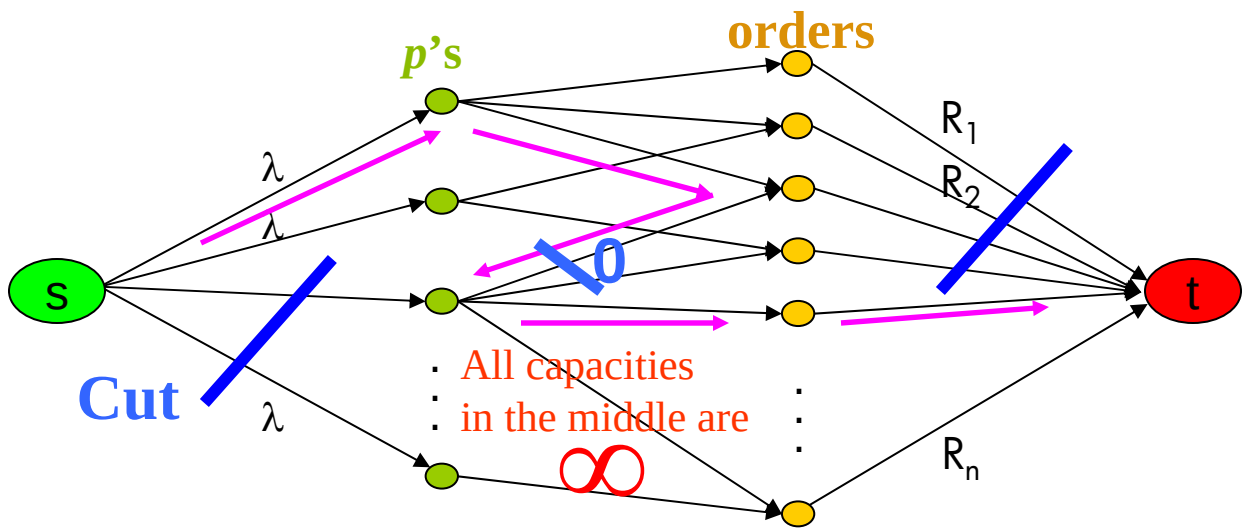
New Balancing Methods for Parametric Maximum-Flow Problems

Convert LP(λ) to A Maximum-Flow Problem

(Balinski 1970, Rhys 1970)

$$p \in o$$

A directed arc from SKU to the order *o* with capacity equal to +infinity



Re-ordering *p*'s in increasing order of flows from *s* and re-ordering orders properly, we can draw the min-cut.

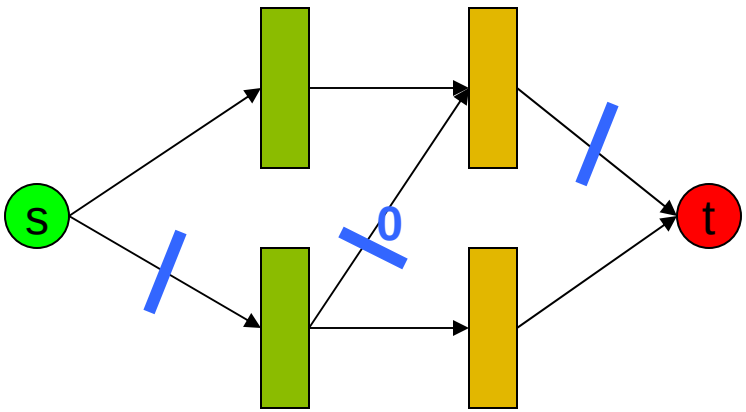
LP(λ):

$$\max \sum_{o \in O} R_o y_o - \lambda \sum_p x_p \longrightarrow \min \sum_{o \in O} R_o (1 - y_o) + \lambda \sum_p x_p$$

$$A = \{s\} \cup \{p \mid flow_{s \to p} = \lambda\} \cup \{o \mid flow_{o \to t} = R_o\}$$

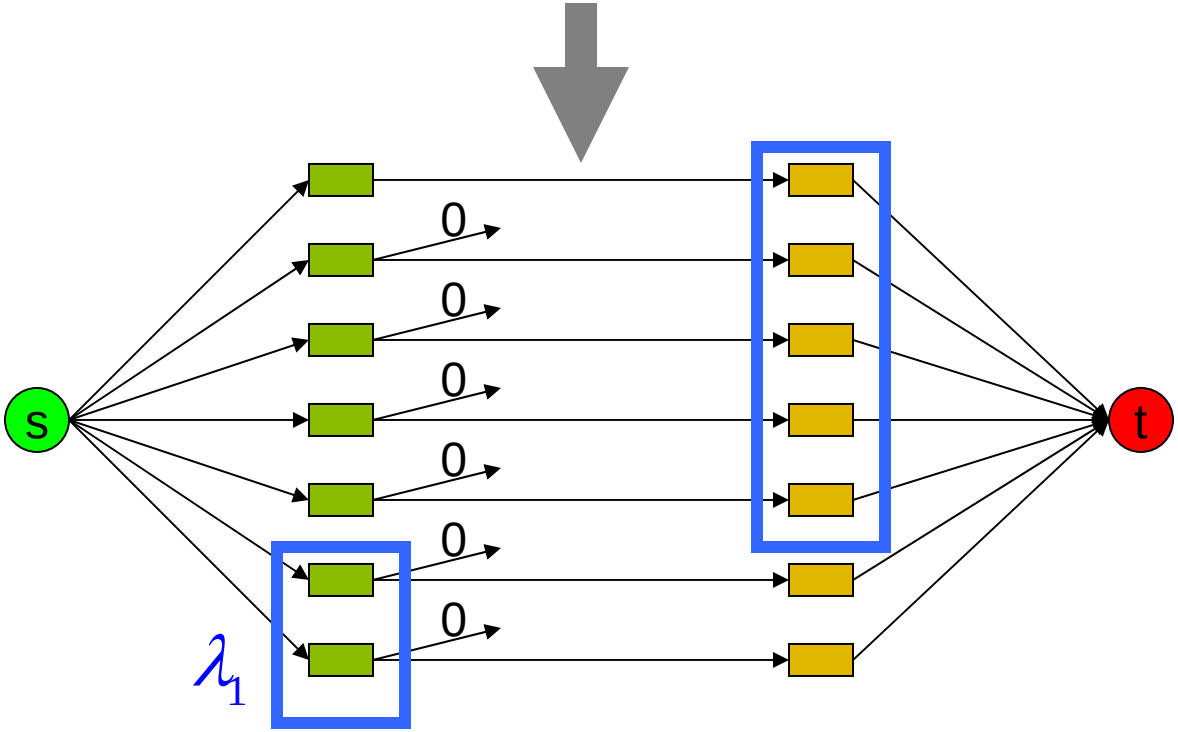
$$Cut = A \mid A^c$$

A View Leading to Balancing Method

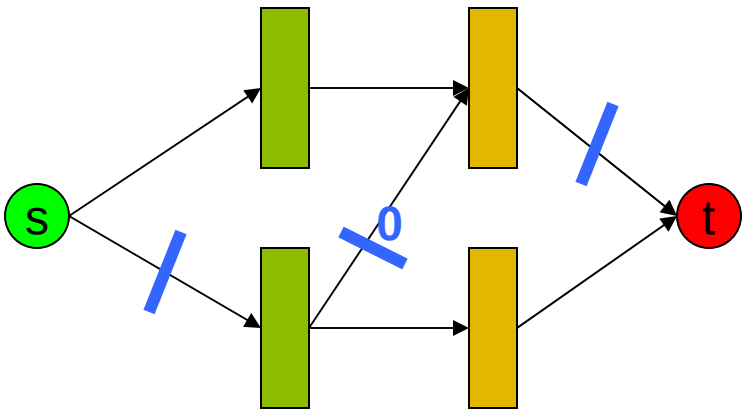


+ Nested Property

$\lambda_1 > \lambda_2 > \lambda_3$

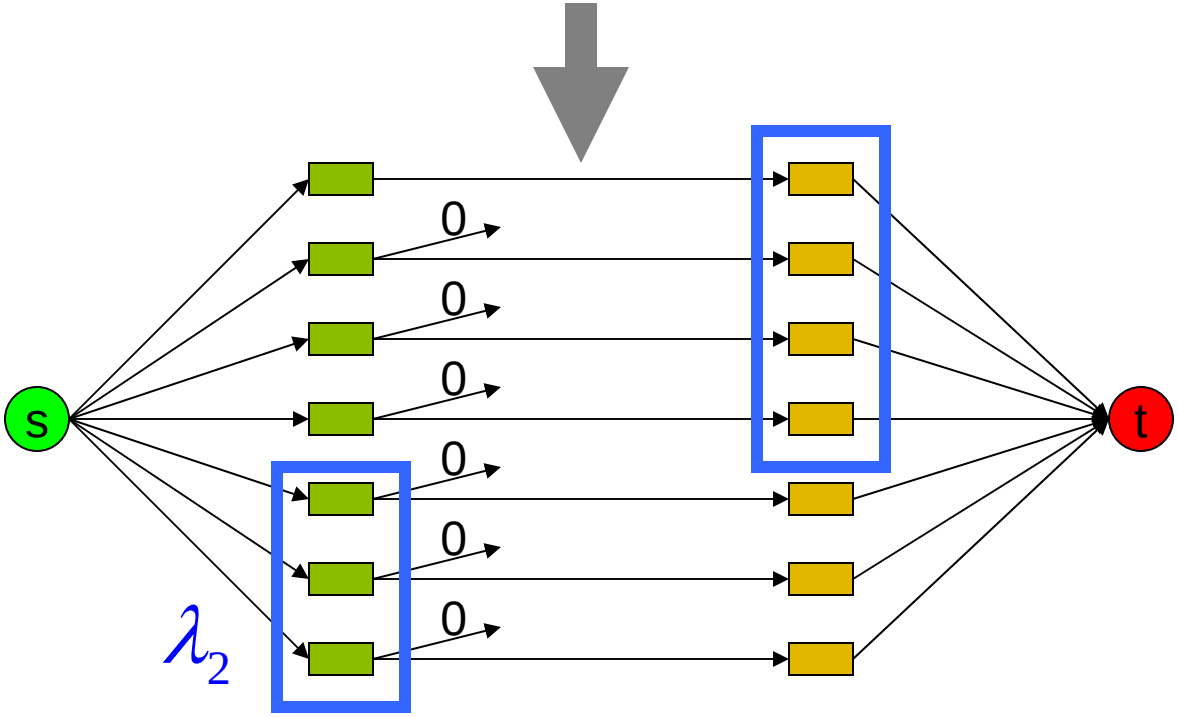


A View Leading to Balancing Method

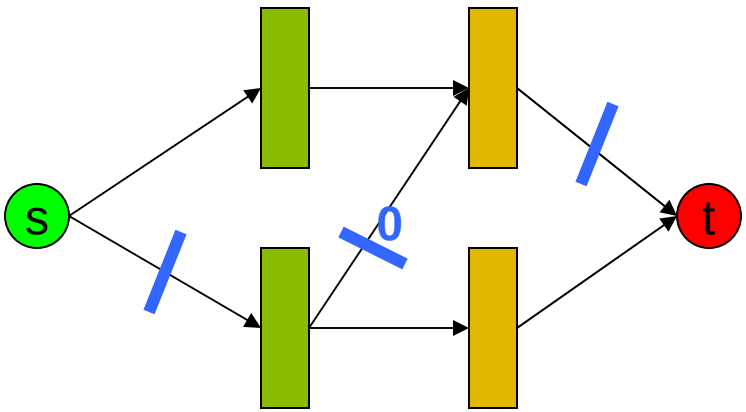


+ Nested Property

$\lambda_1 > \lambda_2 > \lambda_3$



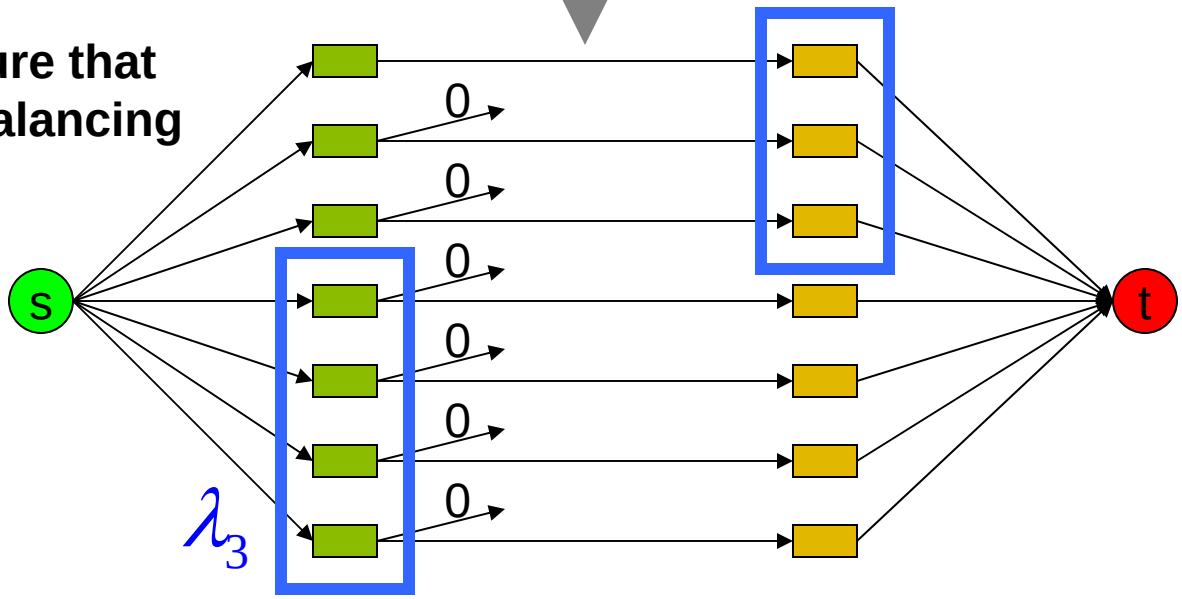
A View That Lead to the Balancing Method



+ Nested Property

$\lambda_1 > \lambda_2 > \lambda_3$

This is the picture that gave the new balancing algorithm.

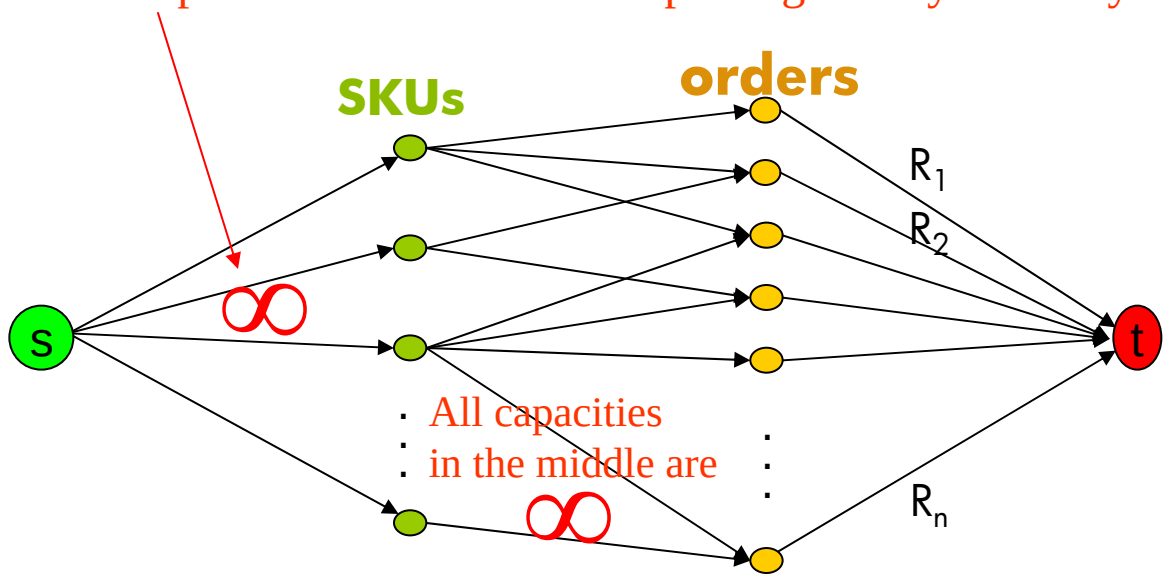


Path Balancing Method

1. Working with a derived non-parametric network
2. Find a special state of flows in the derived network so that
3. Minimum-cuts at all breakpoints of λ can be read from it.

- Example:

Step 1: a derived non-parametric network = replacing all l by +infinity

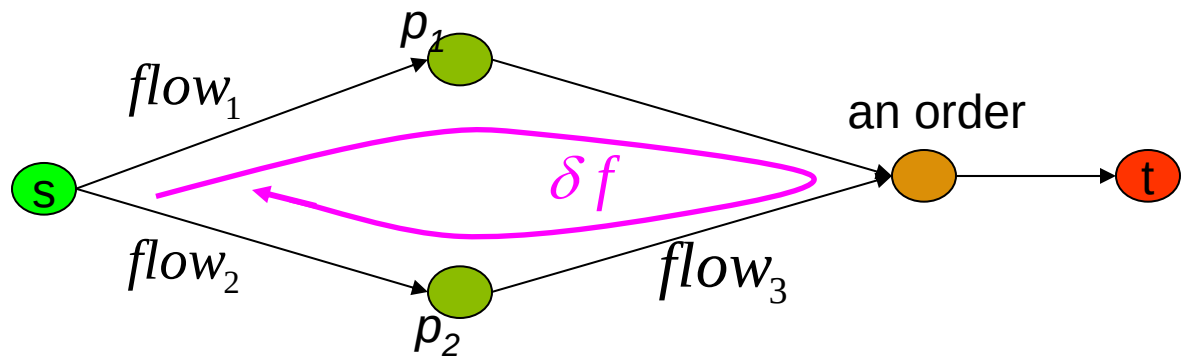


Path Balancing Method (Step 2)

- Find a special state of flows
  - Initialize the flow in the derived network to its maximum-flow (easy in this case).
  - Carry out *balancing* as long as there is work to do.

Path Balancing

For any occurrences of $flow_1 < flow_2$ and $flow_3 > 0$



augmenting a flow $\delta f = \min\{\frac{flow_2 - flow_1}{2}, flow_3\}$

$$flow_1^{new} = flow_1 + \delta f,\ flow_2^{new} = flow_2 - \delta f,\ flow_3^{new} = flow_3 - \delta f$$

After balancing either $flow_1^{new} = flow_2^{new}$ or $flow_3^{new} = 0$

Path Balancing Method (Step 2)

- Convergence of Step 2: The positive sum below is monotone decreasing:

$$\sum_{all\ p's} flow^2_{s\to p}$$

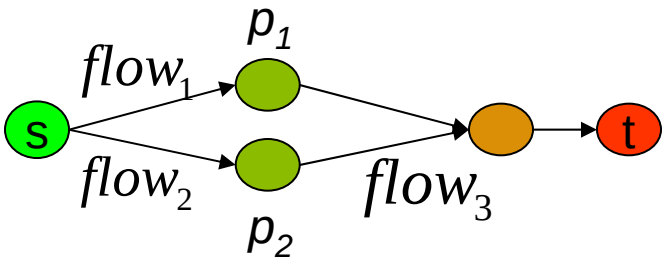
$$(flow_1+\delta f)^2+(flow_2-\delta f)^2 = flow_1^2+flow_2^2-2[(flow_1+flow_2)-\delta f)]\delta f < flow_1^2+flow_2^2$$

(The issue of stop in finite number of steps is addressed later.)

- The end state of step 2

For any loop with $flow_1 \le flow_2$

Either $flow_1 = flow_2$ or $flow_3 = 0$

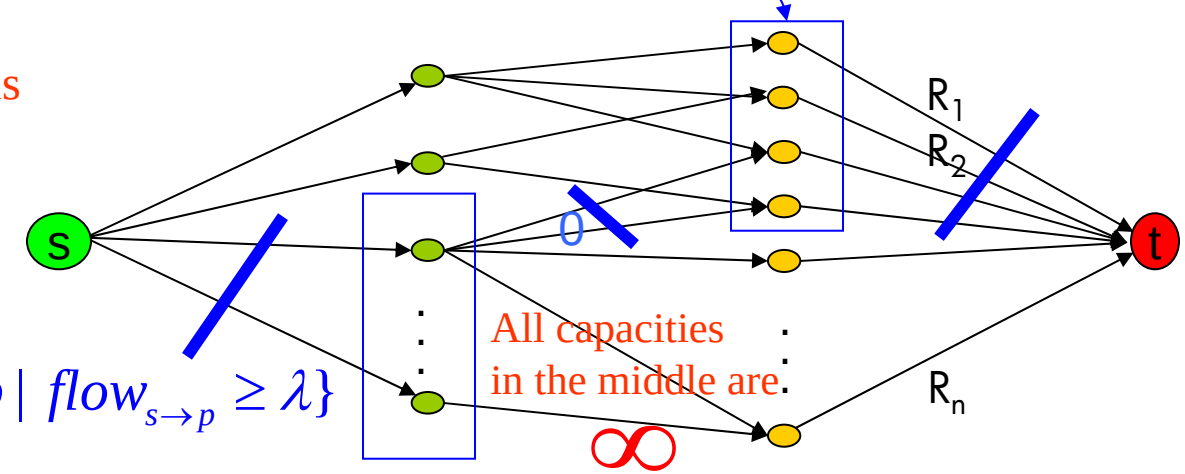


Proof of Minimum-Cuts

(under theoretically sound stopping rule)

- Recover the original network from the derived network with the special state of flows by putting the l capacity back.
- Reduce all $flow_{s \to p}$ that violates the l capacity.
- Rebalance the flows at these *p*'s by reducing out-going flows from them.
- Rebalancing flows at all the orders that are affected by reducing the flows from them to the sink *t*.
- This rebalancing phase will not change any of the out-going flows from the orders in $\{o \mid o \not\subset B_\lambda\}$

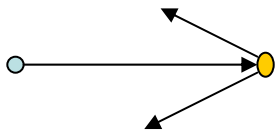
All the steps here are for proof only. None is in the implementation.



$B_\lambda = \{p \mid flow_{s \to p} \geq \lambda\}$

Implementation of Path Balancing

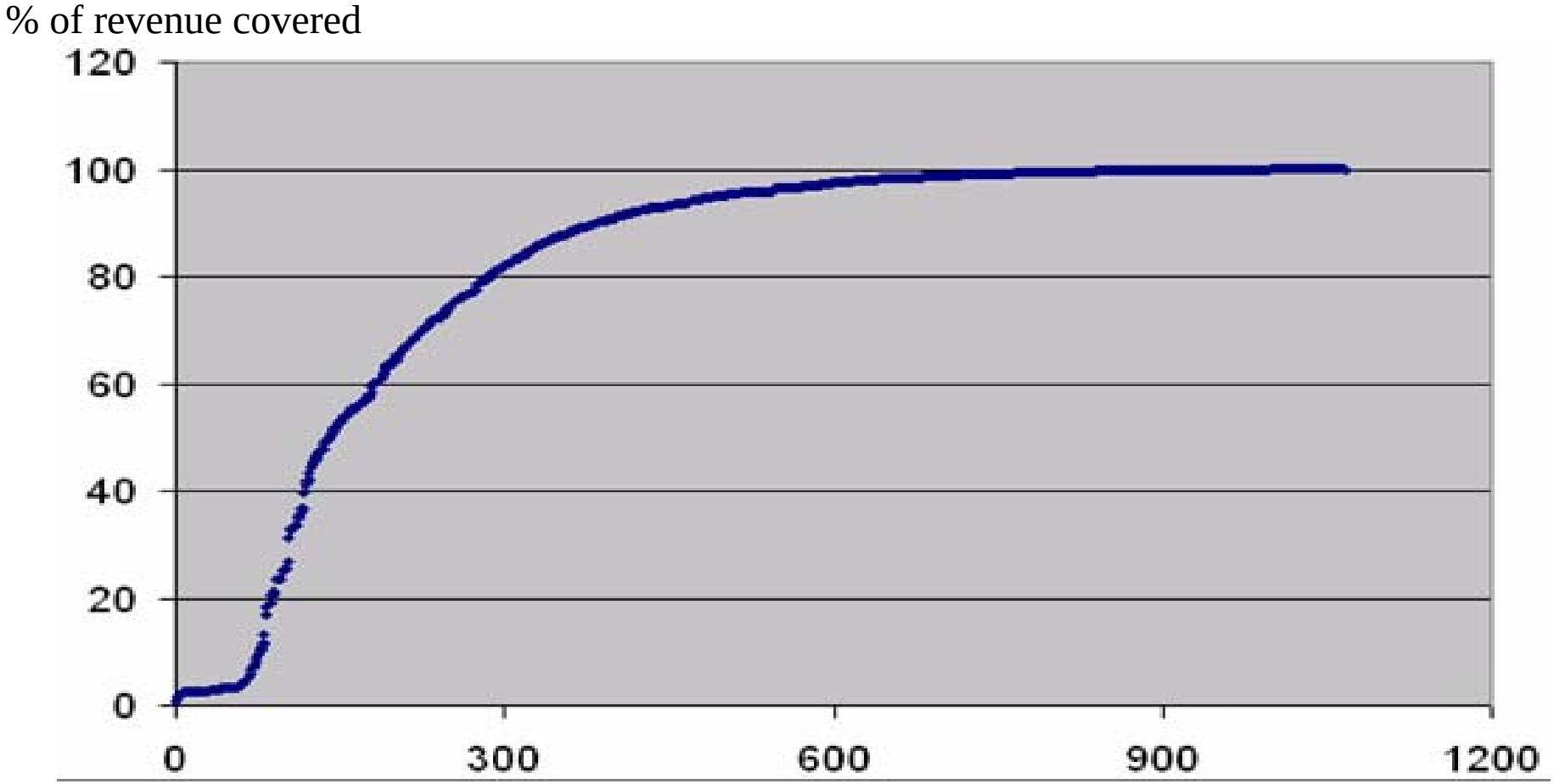
- A round-robin queue is used for scanning all *p*'s
- Balancing at each *p* (to all orders it connects to and all other *p*'s connected that order) as a group



- If no work can be done at a *p*, it is removed from the queue (deactivation of *p*)
- When the queue becomes empty (all *p*'s are deactivated), verification of stopping rule is done. Any *p* that still has work to be done is put back to the round-robin queue.
- Stop if verification pass finds no more work to do.

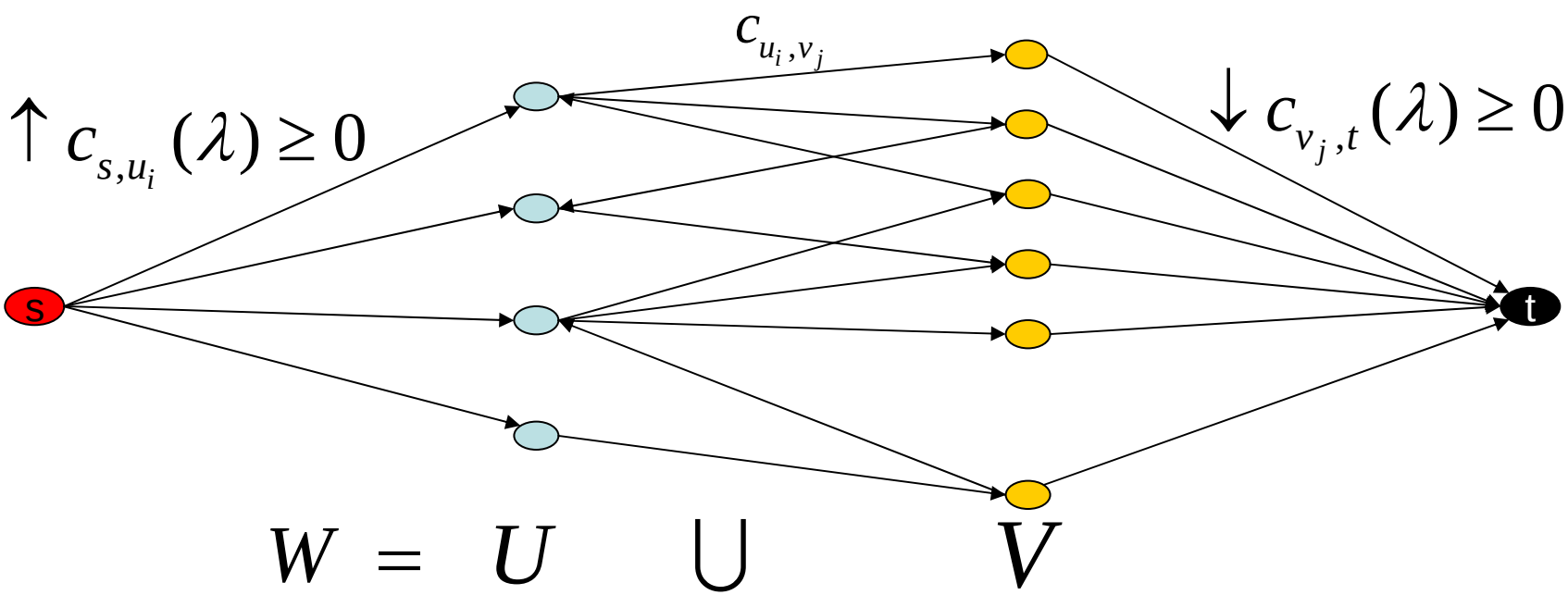
One Curve from Real-World Data

(presented at INFORMS, 4/2005)



The number of dots in this plot is the number of approximate minimum-cuts found.

Path Balancing for Bipartite Monotone Parametric Flow Networks



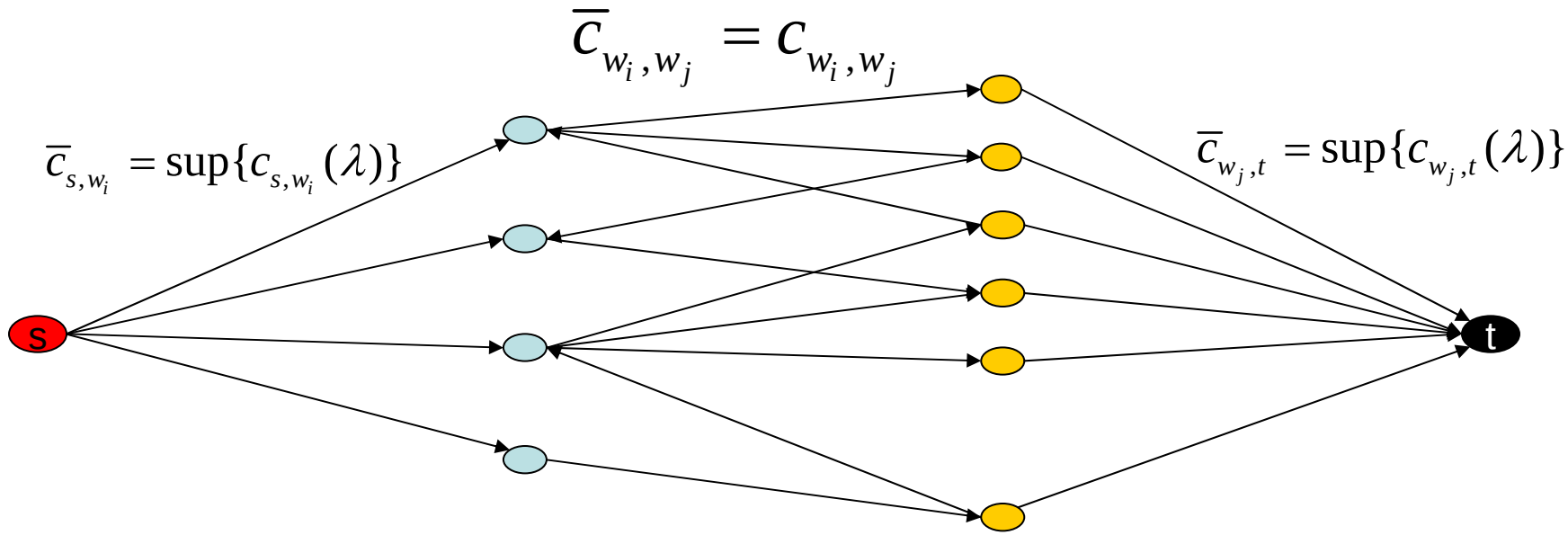
Monotone increasing parametric capacities on the arcs incident to *s*.
Monotone decreasing parametric capacities on the arcs incident to *t*.
Fixed capacities on all other arcs.

} → Minimum-cuts under diff. λ are nested.

Studied before in : G. Gallo, M.D. Grigoriadis & R.E. Tarjan (1989)

It can also be solved using the Path-Balancing algorithm.

Step 1: The Non-Parametric Derived Network



$$W = U \cup V$$

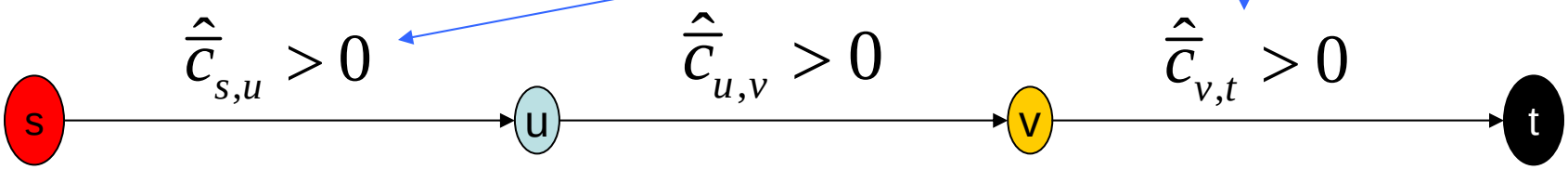
$$\lambda_{s,w_i} = c_{s,w_i}^{-1}(f_{s,w_i}) \qquad \lambda_{w_j,t} = c_{w_j,t}^{-1}(f_{w_j,t})$$

Practical when the inverse functions can be evaluated with acceptable cost.
Example: Piecewise linear approximations can be used.

λ-Directed Simple Residual Paths

st-type: $\lambda_{s,u} < \lambda_{v,t}$ & $\hat{\bar{c}}_{u,v} > 0$

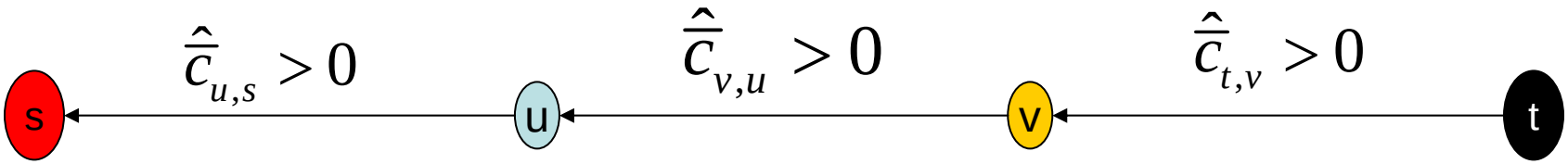
Implied



Residual capacity > 0 $\begin{cases} \hat{\bar{c}}_{s,u} = \bar{c}_{s,u} - \bar{f}_{s,u} = h_{s,u} - c_{s,u}(\lambda_{s,u}) > h_{s,u} - c_{s,u}(\lambda_{v,t}) \geq 0 \\ \hat{\bar{c}}_{v,t} > 0 \text{ can be derived similarly.} \end{cases}$

---

ts-type: $\lambda_{s,u} > \lambda_{v,t}$ & $\hat{\bar{c}}_{v,u} > 0$

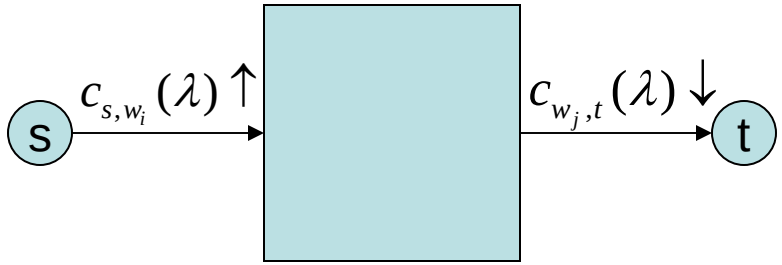


Path-Balancing Algorithm (SPMF)

- Step 2: Initialize all flows to zero. Do the following until there is no more work left:
  - For a λ-directed simple residual path, augment the flow to eliminate it as a λ-directed simple residual path.
  - After augmenting the flow, either the two λ values become the same or the residual capacity of the path becomes zero.
  - Convergence proved in HPL-2004-189
- Step 3: Sort both sides of vertices according to the associated λ-value and scan the vertices in this sorted order to print out all the min-cuts and their associated capacities (which are equal to the value of maximum-flows).

Applying to More General Networks

- SPMF is essentially an augmenting-path algorithm. It augments flows along only λ-directed simple residual paths.
- It naturally extends to more general networks bellow:

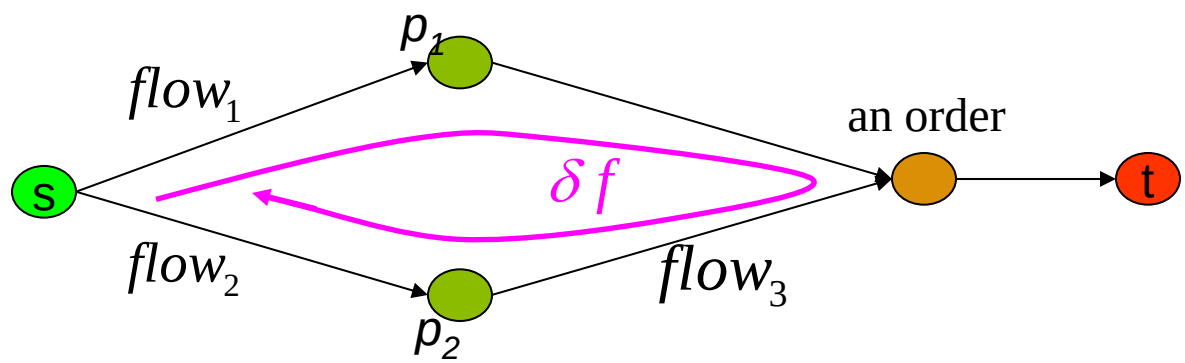


Star-Balancing
(old name: Vertex-Balancing)

Path Balancing Method (Step 2)

refresh memory

For any occurrences of $flow_1 < flow_2$ and $flow_3 > 0$



augmenting a flow $\delta f = \min\{\frac{flow_2 - flow_1}{2}, flow_3\}$

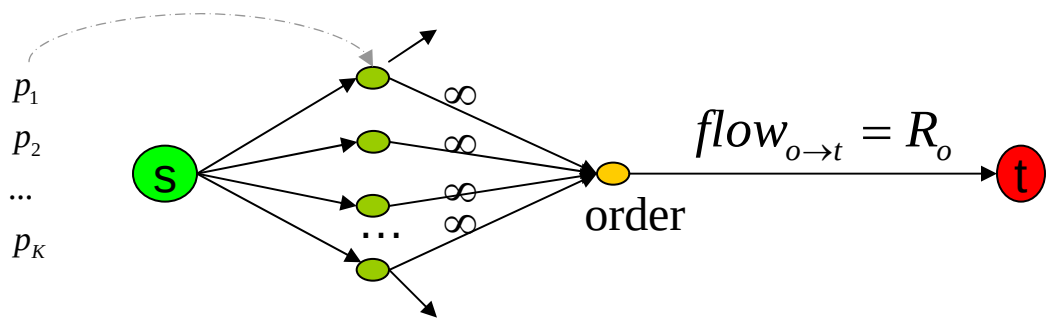
$$flow_1^{new} = flow_1 + \delta f,\ flow_2^{new} = flow_2 - \delta f,\ flow_3^{new} = flow_3 - \delta f$$

Either $flow_1^{new} = flow_2^{new}$ or $flow_3^{new} = 0$

Star-Balancing Algorithm

(was called vertex balancing)

- Looking at one order, there are many loops going through it.
  - Instead of balancing each loop, Star-Balancing redistributes the flows going through the same order so that
  - after the redistribution, no more path balancing involving this order is left to do.

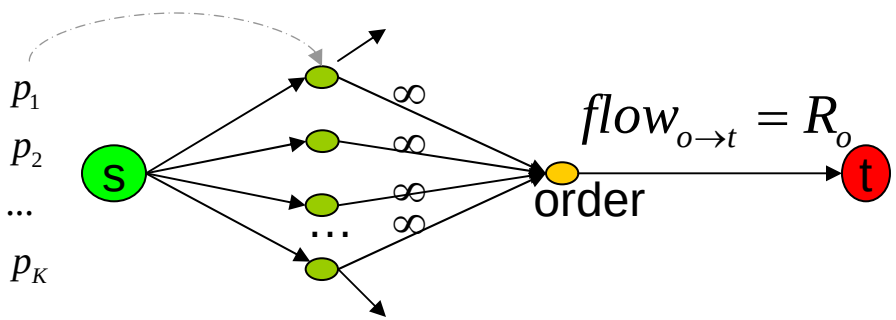


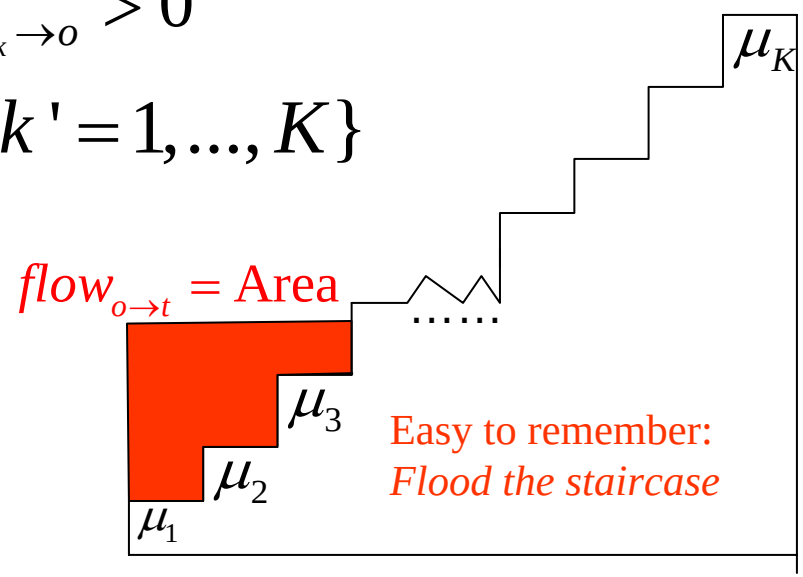
Formulas for Redistribution

- Step 1: taking the flow contribution from *o* out of the current flows $\mu_k = f_{s \to p_k} - f_{p_k \to o}, k = 1, \ldots, K$
- Step 2: sort the remaining flows $\{\mu_k\}_1^K$

  Assume: $\mu_1 \le \mu_2 \le \ldots \le \mu_K$
- Step 3: put the flow contribution from *o* back with a new distribution so that if $f^{new}_{p_k \to o} > 0$

$$\longrightarrow \quad f^{new}_{s \to p_k} = \min\{ f^{new}_{s \to p_{k'}} \mid k' = 1, \ldots, K \}$$





In Technical Report: HPL-2005-121

Comments on Star-Balancing

- Since Star-balancing is an "accelerated path-balancing", it also has monotone convergence.
- Sorting is the most expensive part: $O(K\log(K))$
- After sorting, finding the new distribution and updating the flows is linear in K.
- Comparing with path balancing, the cost of one pass through in this local star is reduced from $O(K^2)$ *to* $O(K\log(K))$

Round-Robin Queue for Orders with Activation/Deactivation

- A round-robin queue is used for all orders.
- If the new distribution of flows is the same as before at an order, it is removed from the queue (deactivation).
- When the queue becomes empty, verification is carried out on all orders using the same star balancing method. If any redistribution of flows happens at an order, it is put back to the queue.
- Stop if no change of flow distribution is found at any orders.

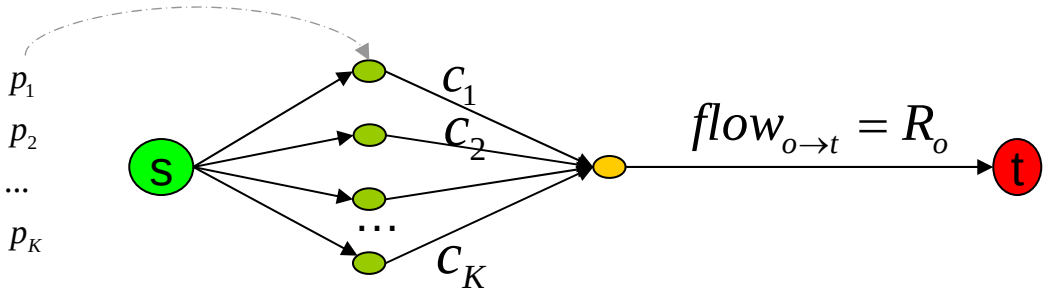
Performance Comparisons

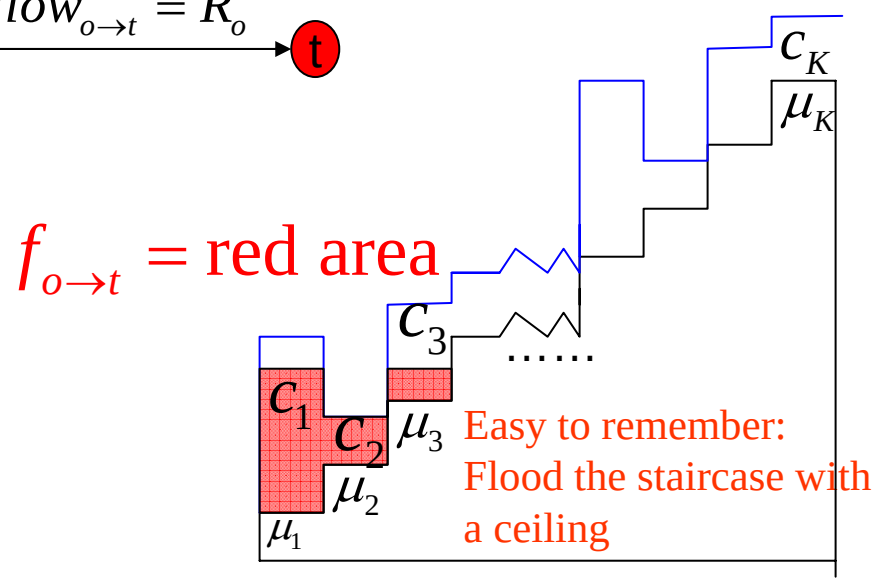
Data Set	$\|P\|$	$\|O\|$	#arcs	#diff. λ values	Path-Balancing (seconds)	Star-Balancing (seconds)
D1	**263**	**39,106**	**416,481**	**189**	**10.7**	**0.67**
D2	**232**	**53,565**	**572,134**	**155**	**8.9**	**0.94**
D3	**344**	**123,933**	**1,280,298**	**222**	**18.9**	**1.88**
D4	**439**	**286,604**	**3,111,773**	**298**	**118**	**9.1**

- All experiments run on the same HP XW8000 Workstation.
- Used a single CPU. All data structures were in memory.
- Timing started right before algorithm's initialization and ended right after algorithm's stop.
- Loading input data from hard disk and printing results to files on disk were excluded from timing.

Star-Balancing (Generalized)

Star-Balancing in a bipartite flow network with finite capacities on the arcs from *p* to *o*.





The capacities form a ceiling above the staircase.
With this picture, writing the formulas for the new flows is a simple exercise.

Star-Balancing Algorithm is extremely simple to understand and to implement compared with previously known algorithms.

Technical Reports on Balancing Method for Parametric Maximum-Flow Problems

By Zhang at el.

- HPL-2004-189: A Simultaneous Parametric Maximum Flow Algorithm that Finds the Complete Chain of Solutions
- HPL-2005-91: A Simultaneous Maximum Flow Algorithm for the Selection Model
- HPL-2005-121: Simultaneous Parametric Maximum Flow Algorithm with Vertex Balancing (Star-Balancing)

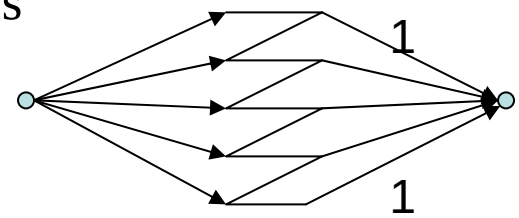
- "Simultaneous" – this kind of algorithms works on all minimum-cuts simultaneously, rather than sequentially.

Bad Case for Balancing Method

- Balancing method worked well with all real-world problems we solved so far, its worst-case bound, derived by Tarjan, is *O(n)* higher than *GGT* that has the best bound. A *"long path"* example (created by Tarjan) shows a bad case:
- With a bad initialization in this long-path network, it takes

  $\Omega(n^3 \log(n))$ to finish balancing.

  n = number of vertices.



- Experimental comparisons confirmed a few things
  - Star-Balancing runs about 5 times faster than GGT on very large real-world data sets tested so far.
  - Bad long path effect is not likely to occur when the revenue has a (random) skewed distribution or when more random arcs are added to the network, instead of just a single long path.
- General comments on worst-case analysis
  - "Conflicting" reports from worst-case analysis and from real testing runs have been observed by other people on a number of different algorithms.
  - Worst-case are often created by using "coincidence". As soon as the quantities (or structure) in the worst-case are replaced by random quantities (or random structures), the chance of hitting a bad case is small.
  - Optimizing algorithm design on the worst-case has its own cost. While it increases the sophistication, it often also increases the complication of the algorithm, and increases the cost on cases that are not the worst.

Other Related Work

- Balancing method is applied to find a minimum-cut of ordinary maximum-flow problems on general networks beyond bipartite.
  - Combined balancing method with pseudo-flows
- Proved sufficient conditions for stopping
- Developed worst-case analysis
- For details, see the paper by Tarjan et al.:
  - R. Tarjan, J. Ward, B. Zhang, Y. Zhou and J. Mao (2006), Balancing Applied to Maximum Network Flow Problems, ESA 2006, 14th Annual European Symposium on Algorithms, Switzerland

References

- M.A. Babenko and A.V. Goldberg (2006), Experimental evaluation of a parametric flow algorithm. Technical report, Microsoft Research.
- M.A. Babenko, J. Derryberry, A.V. Goldberg, R. Tarjan, Y. Zhou (2007), Experimental evaluation of parametric max-flow algorithms, WEA'07
- M.L. Balinski (1970), On a selection problem, Management Science, 17(3):230-231
- B.V. Cherkassky & A.V. Goldberg (1997), On implementing Push-Relabel Method for the Maximum-Flow Problem, Algorithmica, 19:390-410
- L.R. Ford and D.R. Fulkerson (1956), Maximum Flow through a Network, Canadian J. Math. 8:339-404
- G. Gallo, M.D. Grigoriadis & R.E. Tarjan (1989), A Fast Parametric Maximum Flow Algorithm and Applications, SIAM J. Computing, Vol. 18(1):30-55
- A.V. Goldberg and R.E. Tarjan (1988), A New Approach to the Maximum-Flow Problem, J. ACM 35(4):921-940
- D. Gusfield and C. Martel (1992), A fast algorithm for the generalized parametric minimum cut problem and applications, Algorithmica, 7:499-519
- D. Gusfield and E. Tardos (1994), A faster parametric minimum-cut algorithm, Algorithmica 11:278-290
- E. Lawler (1976), Combinatorial Optimization – Networks and Matroids

END