Generalized Bipartite Matching

Peli Teloni

Network Algorithms
$\mu\Pi\lambda\forall$

July 22, 2014

Outline

Real-Life situation

Bob

Real-Life situation

Bob

Real-Life situation

Bob

e-mail from DVD-rental store

Real-Life situation

Bob
wants to watch
a **movie**

e-mail from DVD-rental store

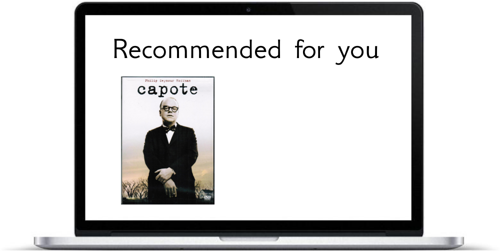
Real-Life situation

Bob

wants to watch

a **movie**

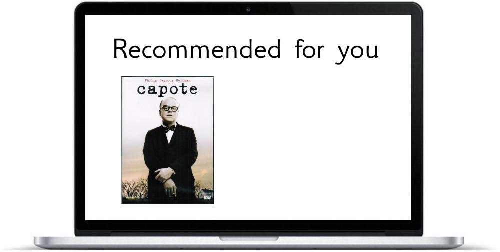
e-mail from DVD-rental store



Real-Life situation

Bob
wants to watch
Capote

e-mail from DVD-rental store



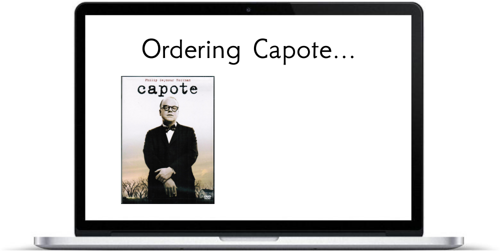
Real-Life situation

Bob

wants to watch

Capote

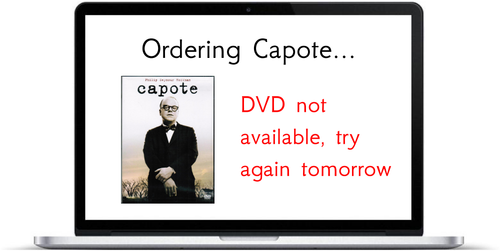
e-mail from DVD-rental store



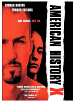
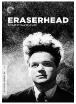
Real-Life situation

Bob
wants to watch
Capote

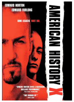
e-mail from DVD-rental store

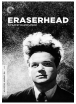


A Simple Recommender System

			
	?	?	2
	4	?	?
	?	5	?
	?	?	?

A Simple Recommender System

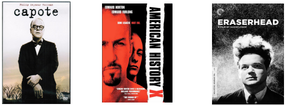




	5	3	2
	4	5	4
	2	5	1
	4	3	1

Step 1: Predict preferences

A Simple Recommender System

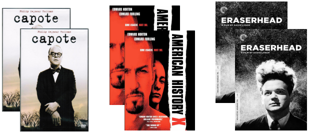


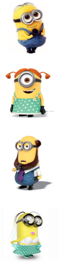
5	3	2
4	5	4
2	5	1
4	3	1

Step 1: Predict preferences

Step 2: Recommend preferred movies

A Simple Recommender System



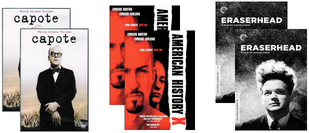


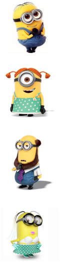
5	3	2
4	5	4
2	5	1
4	3	1

Step 1: Predict preferences

Step 2: Recommend preferred movies

A Simple Recommender System



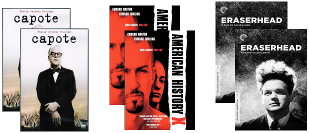


5	3	2
4	5	4
2	5	1
4	3	1

Step 1: Predict preferences

Step 2: Recommend preferred movies

A Simple Recommender System

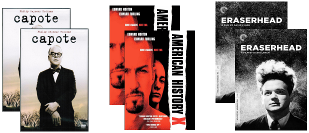


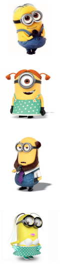
5	3	2
4	5	4
2	5	1
4	3	1

Step 1: Predict preferences

Step 2: Recommend preferred movies *under constraints*

A Simple Recommender System



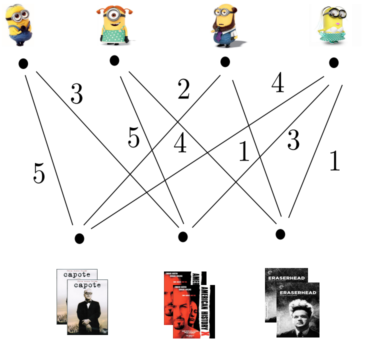


5	3	2
4	5	4
2	5	1
4	3	1

Step 1: Predict preferences

Step 2: Recommend preferred movies *under constraints*

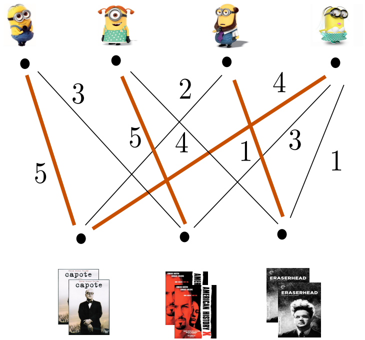
What we just saw



$$1 \leq d_u \leq 1$$

$$0 \leq d_v \leq 2$$

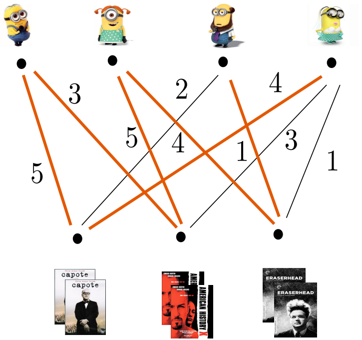
What we just saw



$$1 \leq d_u \leq 1$$

$$0 \leq d_v \leq 2$$

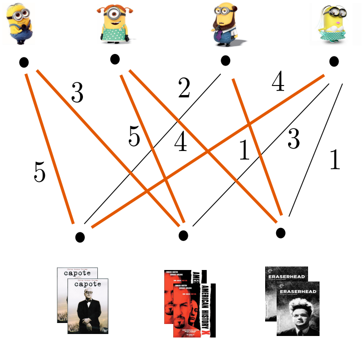
What we just saw



$$1 \leq d_u \leq 2$$

$$0 \leq d_v \leq 2$$

What we just saw



$$1 \leq d_u \leq 2$$

$$0 \leq d_v \leq 2$$

Generalized Bipartite Matching

Input: a bipartite weighted graph $G = (U, V, E)$ with degree constraints

Output: a max-weight subset of edges that satisfy these constraints

Solving GBM

- Optimal solution in poly-time
  - max-flow techniques
  - linear programming formulation

Solving GBM

- Optimal solution in poly-time
  - max-flow techniques
  - linear programming formulation
- solvers in practice (e.g. Gurobi)
  - behave well for up to medium size instances
  - break down to large instances
  - at present: Netflix has 20M users, 10k's of movies

Solving GBM

- Optimal solution in poly-time
  - max-flow techniques
  - linear programming formulation
- solvers in practice (e.g. Gurobi)
  - behave well for up to medium size instances
  - break down to large instances
  - at present: Netflix has 20M users, 10k's of movies
- In this paper: near-optimal solution
  - poly-log time
  - strong approximation guarantees
  - first distributed algorithm

Overview for GBM_ϵ

Definition of GBM_ϵ

Given $\epsilon > 0$ and an instance of GBM, if GBM is feasible then find $\bar{E} \subseteq E$ s.t.:

$$\lfloor (1-\epsilon) l(v) \rfloor \leq |\bar{E}_v| \leq \lceil (1+\epsilon) b(v) \rceil \quad \forall v \in U \cup V$$

$$\sum_{e \in \bar{E}} w(e) \geq (1-\epsilon)\text{OPT}$$

Overview for GBM_ϵ

Definition of GBM_ϵ

Given $\epsilon > 0$ and an instance of GBM, if GBM is feasible then find $\overline{E} \subseteq E$ s.t.:

$$\lfloor (1-\epsilon)l(v) \rfloor \leq |\overline{E}_v| \leq \lceil (1+\epsilon)b(v) \rceil \quad \forall v \in U \cup V$$

$$\sum_{e \in \overline{E}} w(e) \geq (1-\epsilon)\text{OPT}$$

Step 1: LP relaxation

- outputs "edge probabilities"
- mixed packing-covering LP
- distributed approximation solver

Overview for GBM$_\epsilon$

Definition of GBM$_\epsilon$

Given $\epsilon > 0$ and an instance of GBM, if GBM is feasible then find $\overline{E} \subseteq E$ s.t.:

$$\lfloor (1-\epsilon) l(v) \rfloor \leq |\overline{E}_v| \leq \lceil (1+\epsilon) b(v) \rceil \quad \forall v \in U \cup V$$

$$\sum_{e \in \overline{E}} w(e) \geq (1-\epsilon)\text{OPT}$$

Step 1: LP relaxation

- outputs "edge probabilities"
- mixed packing-covering LP
- distributed approximation solver

Step 2: Rounding

- outputs actual matching
- distributed dependent
- keeps approx. guarantees

Outline

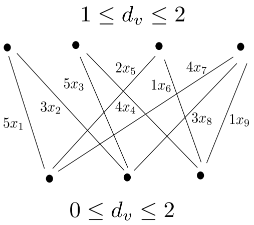
MPC-LP

$$\begin{aligned} \max\ & w^T x \\ \text{s.t.}\ & Px \le p \\ & Cx \ge c \\ & x \ge 0 \end{aligned}$$

- subclass of LPs
- non negative coefficients/variables
- facility location, circuit routing etc.

MPC-LP

$$\begin{aligned} \text{max } & w^T x \\ \text{s.t. } & Px \le p \\ & Cx \ge c \\ & x \ge 0 \end{aligned}$$

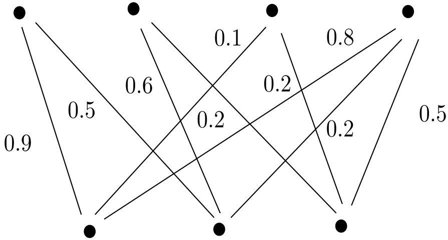


- subclass of LPs
- non negative coefficients/variables
- facility location, circuit routing etc.
- LP_{GBM} is an example of LP_{MPC}
  - constraints: Ms
  - variables and weights: Bs

$$\begin{aligned} \text{max } & \sum_{e \in E} w(e) x_e \\ \text{s.t. } & \sum_{e \in E_v} x_e \le b(v) \\ & \sum_{e \in E_v} x_e \ge l(v) \\ & x_e \in \{0, 1\} \end{aligned}$$

MPCSolver

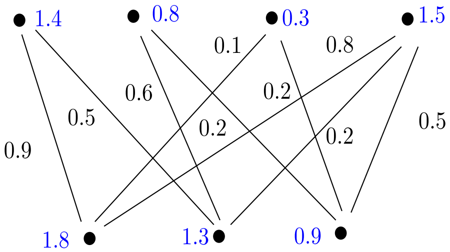
$$1 \le d_v \le 2$$



$$0 \le d_v \le 2$$

MPCSolver

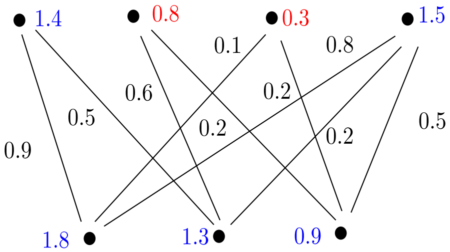
$$1 \leq d_v \leq 2$$



$$0 \leq d_v \leq 2$$

MPCSolver

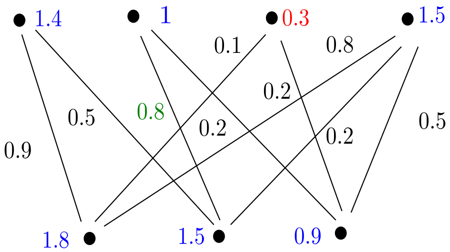
$$1 \leq d_v \leq 2$$



$$0 \leq d_v \leq 2$$

MPCSolver

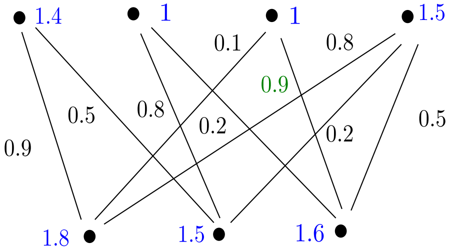
$$1 \leq d_v \leq 2$$



$$0 \leq d_v \leq 2$$

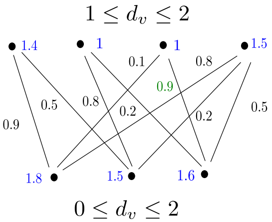
MPCSolver

$$1 \leq d_v \leq 2$$



$$0 \leq d_v \leq 2$$

MPCSolver



repeat:

compute $y_i(x) = \exp(\mu(P_i x - 1))$ for $i \in [m]$

compute $z_i(x) = \exp(\mu(1 - C_i x))$ for $i \in [k]$

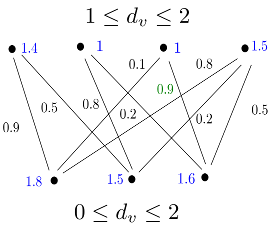
for $j = 1, \ldots, n$

if $\frac{P_j^T y}{C_j^T z} \le 1 - a$ then $x_j = \max\{x_j(1+\beta), \delta\}$

if $\frac{P_j^T y}{C_j^T z} \ge 1 + a$ then $x_j = x_j(1-\beta)$

until convergence

MPCSolver



repeat:

compute $y_i(x) = \exp(\mu(P_i x - 1))$ for $i \in [m]$

compute $z_i(x) = \exp(\mu(1 - C_i x))$ for $i \in [k]$

for $j = 1, \ldots, n$

if $\frac{P_j^T y}{C_j^T z} \leq 1 - a$ then $x_j = \max\{x_j(1+\beta), \delta\}$

if $\frac{P_j^T y}{C_j^T z} \geq 1 + a$ then $x_j = x_j(1-\beta)$

until convergence

$\widetilde{O}\left(\frac{1}{\epsilon^5} \ln^3(kmMnx_{\max})\right)$ rounds

Practical MPCSolver

- Fast convergence: poly-log rounds
- Almost feasible: constraints satisfied up to $(1 \pm \epsilon)$
- Easy to implement: matrix-vector operations

Practical MPCSolver

- Fast convergence: poly-log rounds
- Almost feasible: constraints satisfied up to $(1 \pm \epsilon)$
- Easy to implement: matrix-vector operations
- Parallelization
    - Shared-memory: straightforward
    - Shared-nothing: clever data partitioning

Practical MPCSolver

- Fast convergence: poly-log rounds
- Almost feasible: constraints satisfied up to $(1 \pm \epsilon)$
- Easy to implement: matrix-vector operations
- Parallelization
    - Shared-memory: straightforward
    - Shared-nothing: clever data partitioning
- Near-optimality: objective at least $(1 - \epsilon)$OPT
    - compute bounds on objective
        - $\lambda_{\min}$ (only covering)
        - $\lambda_{\max}$ (only packing)
    - add constraint $w^T x \geq \lambda$
    - binary search for λ
    - at most $\log_2 \log_{1-\epsilon} \left(\frac{\lambda_{\min}}{\lambda_{\max}} \right)$ steps

Outline

GBM Rounding

Input: a solution of GBM_ϵ (near-optimal, ϵ-feasible, fractional)
Output: an integral solution that preserves

1. ϵ-feasibility
2. near-optimality

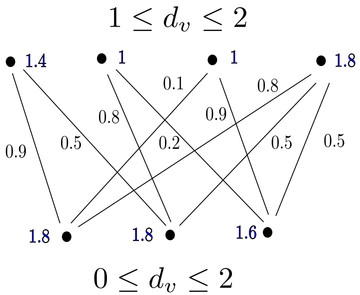
GBM Rounding

Input: a solution of GBM_ϵ (near-optimal, ϵ-feasible, fractional)
Output: an integral solution that preserves

1. ϵ-feasibility
2. near-optimality

Idea 1: independent rounding

- Satisfies (2) in expectation
- Violates (1)



GBM Rounding

Input: a solution of GBM_ϵ (near-optimal, ϵ-feasible, fractional)

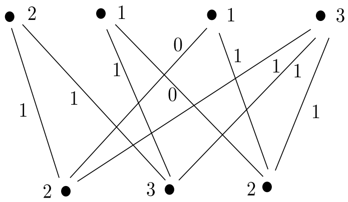
Output: an integral solution that preserves

1. ϵ-feasibility
2. near-optimality

Idea 1: independent rounding

- Satisfies (2) in expectation
- Violates (1)

$$1 \leq d_v \leq 2$$



$$0 \leq d_v \leq 2$$

GBM Rounding

Input: a solution of GBM_ϵ (near-optimal, ϵ-feasible, fractional)

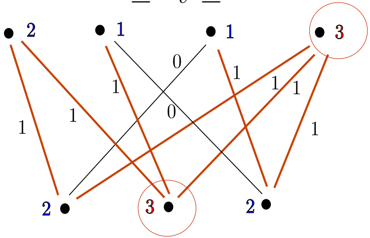
Output: an integral solution that preserves

1. ϵ-feasibility
2. near-optimality

Idea 1: independent rounding

- Satisfies (2) in expectation
- Violates (1)

$$1 \leq d_v \leq 2$$



$$0 \leq d_v \leq 2$$

GBM Rounding

Input: a solution of GBM_ϵ (near-optimal, ϵ-feasible, fractional)

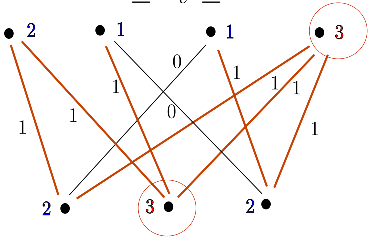
Output: an integral solution that preserves

1. ϵ-feasibility
2. near-optimality

Idea 1: independent rounding

- Satisfies (2) in expectation
- Violates (1)
- need for dependent rounding

$$1 \leq d_v \leq 2$$

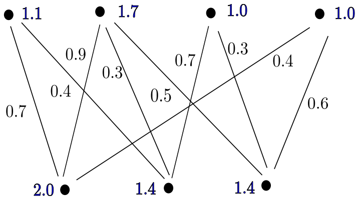


$$0 \leq d_v \leq 2$$

Dependent Rounding

Warm up

Randomized *sequential* Rounding [Gandhi et al. 2006]

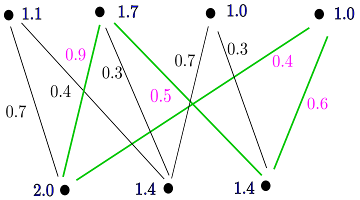


Dependent Rounding

Warm up

Randomized *sequential* Rounding [Gandhi et al. 2006]

1. find a fractional cycle or maximal path

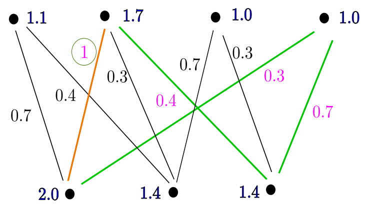


Dependent Rounding

Warm up

Randomized *sequential* Rounding [Gandhi et al. 2006]

1. find a fractional cycle or maximal path
2. round at least 1 edge on the cycle/path

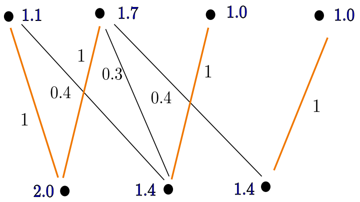


Dependent Rounding

Warm up

Randomized *sequential* Rounding [Gandhi et al. 2006]

1. find a fractional cycle or maximal path
2. round at least 1 edge on the cycle/path
3. repeat

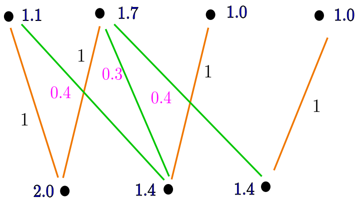


Dependent Rounding

Warm up

Randomized *sequential* Rounding [Gandhi et al. 2006]

1. find a fractional cycle or maximal path
2. round at least 1 edge on the cycle/path
3. repeat

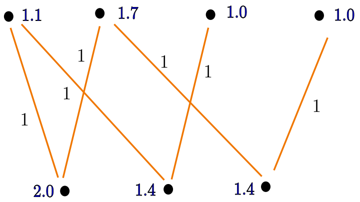


Dependent Rounding

Warm up

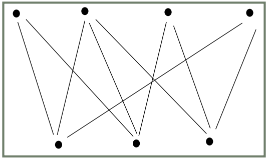
Randomized *sequential* Rounding [Gandhi et al. 2006]

1. find a fractional cycle or maximal path
2. round at least 1 edge on the cycle/path
3. repeat



Distributed Rounding

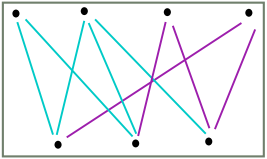
- Partition edges to compute nodes
  - size of each node: number of vertices
  - each node: same number of edges



Full graph

Distributed Rounding

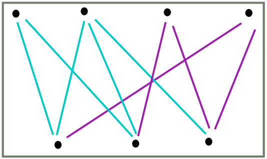
- Partition edges to compute nodes
  - size of each node: number of vertices
  - each node: same number of edges



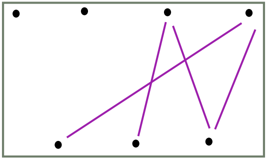
Full graph

Distributed Rounding

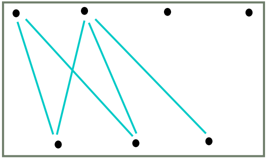
- Partition edges to compute nodes
  - size of each node: number of vertices
  - each node: same number of edges
- local cycle $\Rightarrow$ global cycle
- local maximal path $\not\Rightarrow$ global maximal path



Full graph



Node 1



Node 2

DDRounding

1. Partition edges uniformally (fractional only)
2. Process local cycles
   - k compute nodes, m vertices $\Rightarrow O(mk)$ edges left
3. merge remaining fractional edges (conceptually)
4. Repeat until graph small
5. Run sequential version (cycles, max. paths)

DDRounding

1. Partition edges uniformally (fractional only)
2. Process local cycles
   - k compute nodes, m vertices $\Rightarrow O(mk)$ edges left
3. merge remaining fractional edges (conceptually)
4. Repeat until graph small
5. Run sequential version (cycles, max. paths)

$$\text{time: } O\left(r^{2+\gamma}\lceil c/\gamma\rceil\right) \begin{cases} n = r^{1+c} & \text{number of edges} \\ \\ \eta = O(r^{1+\gamma}) & \text{size of node} \end{cases}$$

DDRounding in Practice

- Edges already partitioned by MPCSolver
- Empirical: most work done in first iteration
- Scales nicely
- Further communication improvement:
  - halving available nodes at each iteration
  - so, e.g. only odd nodes send their data
- Further time improvement:
  - a node performs cycle detection using DFS
  - when cycle hit, an edge is rounded
  - restart DFS?
  - no! decompose cycle C to paths C_1 and C_2 (reverse)
  - replace DFS stack of C with $\max C_1, C_2$
  - mark node if no cycle found

Outline

References

Rainer Gemulla.
presentation.
2013.

Faraz Makari Manshadi, Baruch Awerbuch, Rainer Gemulla, Rohit Khandekar, Julián Mestre, and Mauro Sozio.
A distributed algorithm for large-scale generalized matching.
Proc. VLDB Endow., 2013.

Thank you!