Online Allocation Case Study: Kidney Exchange with Compatible Pairs

Sanmay Das

Optimization & Learning Approaches to Resource Allocation for Social Good (Tutorial @ AAMAS 2019)

Kidney Exchange in Practice

Problems

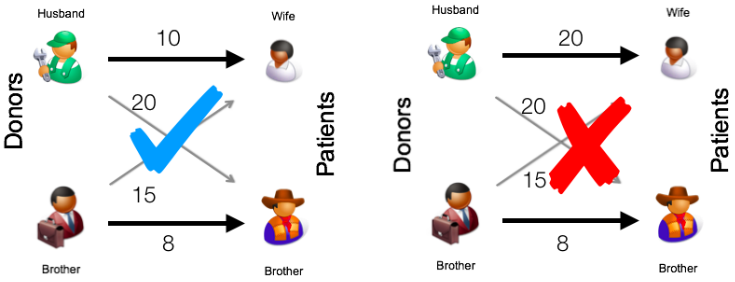
- A raft of coordination problems
- Exchange fragmentation

Parts of the solution

- More pooling of pairs (national/international exchanges)
- Desensitization / ABO incompatible transplants
- Today: **Incorporate compatible pairs into exchanges** (Z. Li et al, *EC 2019*, forthcoming)

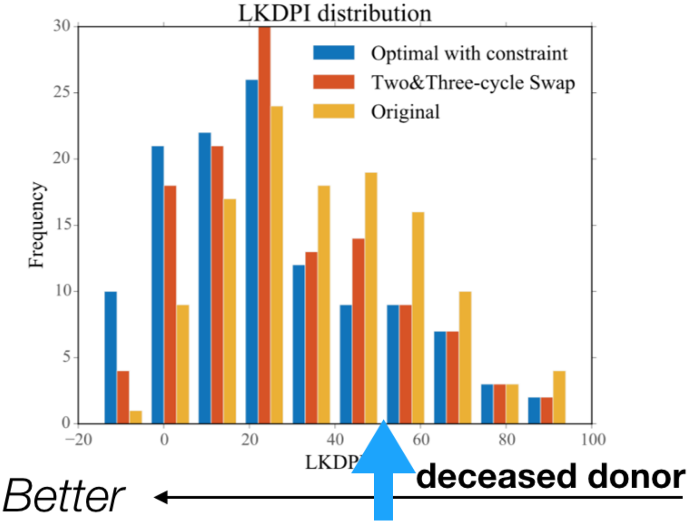
Incorporating Compatible Pairs

- Why would a compatible pair want to enter the exchange?
  - Get a better kidney (Gentry et al *Am. J. Transplantation, 2007*, Anshelevich, Das, & Naamad, *SAGT 2009, JAAMAS 2013*)



Single Center Analysis

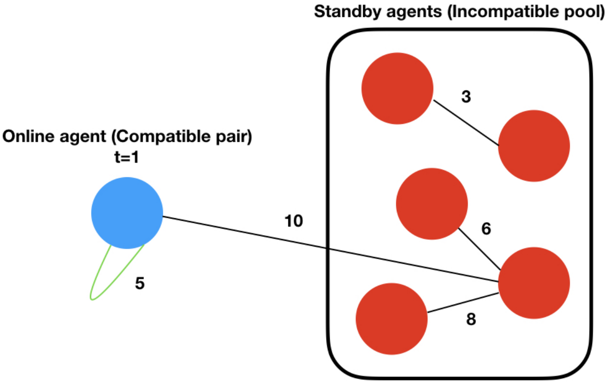
- De-identified data from 2014 - 2016
  - ◇ All donor and recipient characteristics for calculating LKDPI (and hence, graft survival)



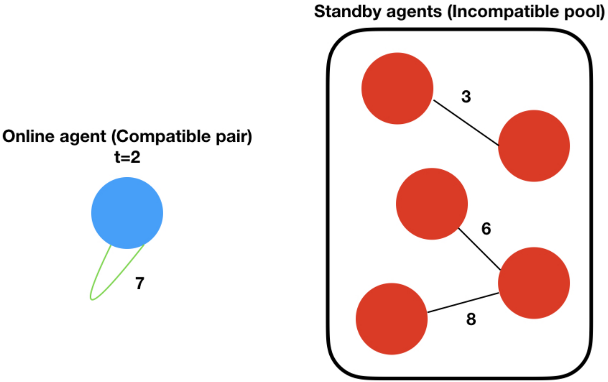
Dynamic Matching

- Compatible pairs may not be willing to wait any longer than necessary
- Also debate in the literature about the value of patience regardless (Akbarpour, S. Li, & Gharan *EC 2014, J. Pol. Econ. Forthcoming*; Anderson et al *SODA 2015, Operations Res. 2017*; Z. Li et al *AMMA 2015, IJCAI 2018*)
- New model: Impatient compatible pairs and a pool of patient incompatible pairs

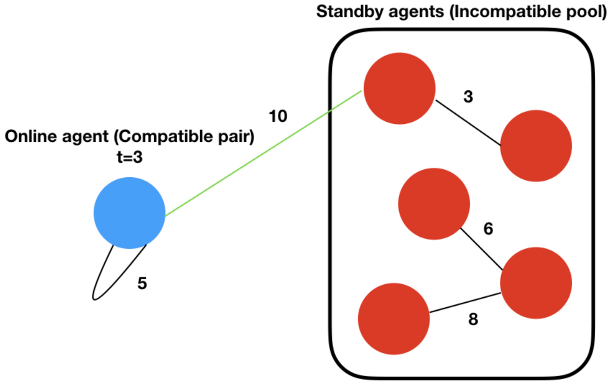
Hybrid Static-Dynamic Matching Model



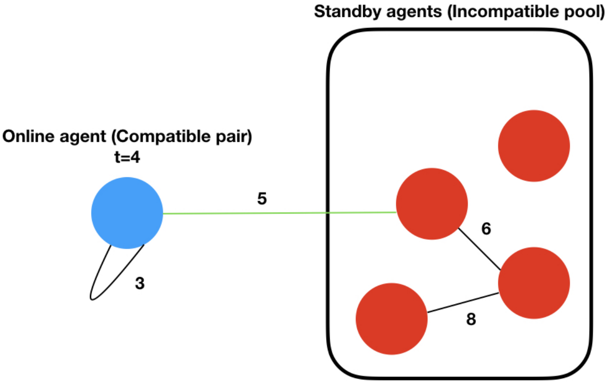
Hybrid Static-Dynamic Matching Model



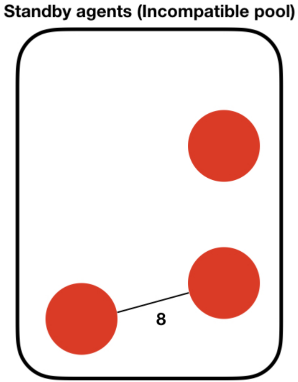
Hybrid Static-Dynamic Matching Model



Hybrid Static-Dynamic Matching Model



Hybrid Static-Dynamic Matching Model



Algorithmic Approach

Most approaches in dynamic settings are based on either greedy or batching mechanisms. We consider a relaxed IP formulation.

Algorithmic Approach

Most approaches in dynamic settings are based on either greedy or batching mechanisms. We consider a relaxed IP formulation.

An Oracle for 2-Matching

$$\max \sum_{n=1}^{N} \sum_{i=0}^{I} w_{n,i} x_{n,i}$$

$$\text{s.t.} \quad \sum_{i=0}^{I} x_{n,i} \leq 1, \forall n \in [T]$$

$$\sum_{n=1}^{N} x_{n,i} + \sum_{j=1}^{I} x_{T+i,j} \leq 1, \forall i \in [I]$$

$$x_{n,i} \in \{0,1\}, \forall n \in [N], \forall i \in [I]^{*}$$

Weights (match quality): $w_{n,i}$

Match variables: $x_{n,i}$

Online agents (compatible pairs): $[T]$

Standby agents (incompatible pairs): $[I]$

All agents: $[N]$

Each online agent matches either with itself (i=0) or with a standby agent (i>0)

Each standby agent matches with exactly one other agent

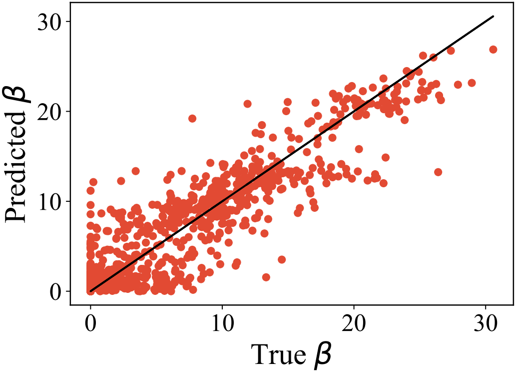
Dual Formulation and the ODASSE Algorithm

$$
\begin{aligned}
\min \quad & \sum_{t=1}^{T} \alpha_t + \sum_{i=0}^{I} \beta_i \\
\text{s.t.} \quad & w_{t,i} - \alpha_t - \beta_i \leq 0, \forall t \in [T], i \in [I]^* \\
& w_{t+j,i} - \beta_j - \beta_i \leq 0, \forall i \in [I], j \in [I] \\
& \alpha_t, \beta_i \geq 0, \forall t \in [T], i \in [I] \\
& \beta_0 = 0
\end{aligned}
$$

- α_t, β_i can be interpreted as estimated values (*shadow survival estimates*) of compatible pairs and incompatible pairs respectively.
- Given optimal β_i^* we can derive the online assignment rule $i^* = \text{argmax}_i \{w_{t,i} - \beta_i^*\}$ (*Online Dual Assignment Using Shadow Survival Estimates*).

Estimating β_i^*

- Run many simulations and get β_i^* values
- Train a linear model on
  - Demographic information of an incompatible pair
  - Initial graph state of incompatible pairs (β_i value when solving the dual on just the incompatible pool).
- Predicted vs. true β^* values.



Measuring the Impact

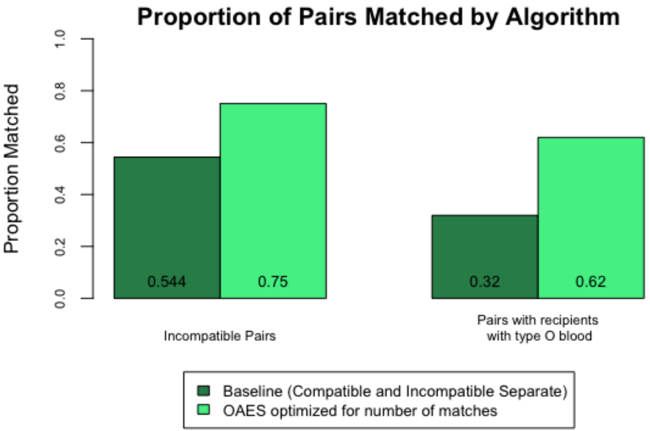
- Including compatible pairs to thicken the exchange with incompatible pairs
  - Increase in the number of matches for incompatible pairs (quantity)
  - Increase in the expected graft survival for compatible pairs (quality)

Results: Potential Social Impact

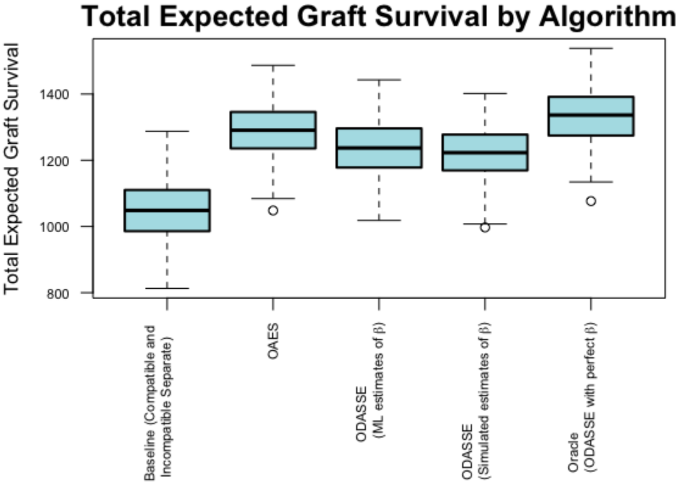
	Baseline	OAES	ODASSE	Oracle
Matched proportion of incompatible pairs	54.4%	74.6%	70.6%	76.0%
Expected graft survival of compatible pairs	9.6	11.1	11.2	11.4
Expected graft survival of incompatible pairs	10.4	9.8	9.6	10.0

OAES (Online allocation via exhaustive search) solves an IP *each time* but only performs the match recommended for the online/impatient agent.

Results: Fairness (O types)



Results: Algorithms

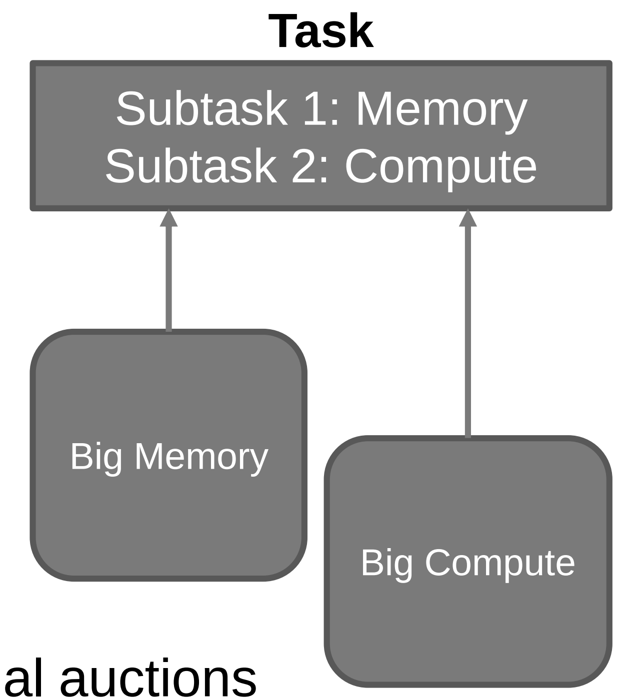


Directions

- Methodological:
  - A model with real weights for weighted matching algorithms to work on!
  - A new hybrid static-dynamic matching model.
  - Online primal-dual + learning algorithm
- Practical:
  - Embed with the surgical team for weekly intake meetings
  - Track waiting times and qualities
  - Implement weighted allocation mechanism in a single center?

CASE STUDY: LEARNING TO MATCH & PACK

BEYOND TRADITIONAL MATCHING



Matching is a special case of the set packing problem

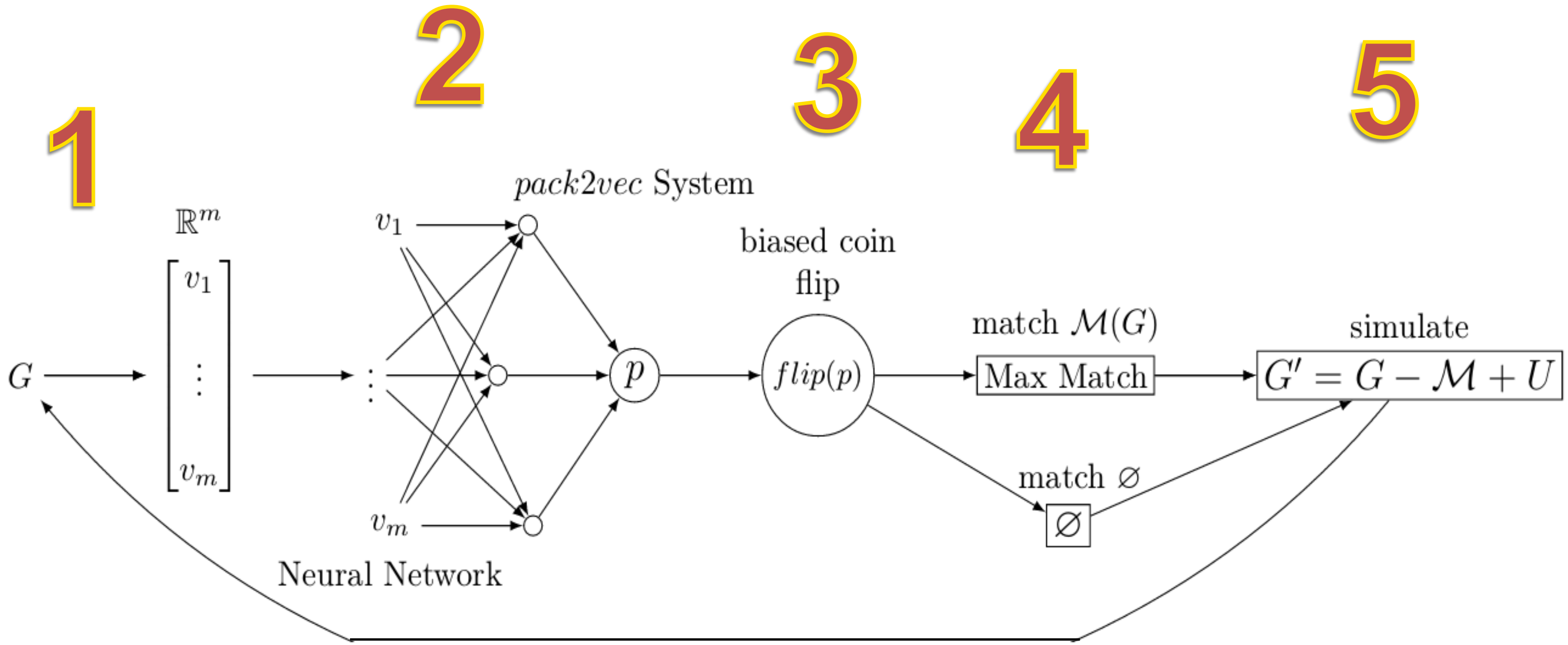
Variants of online set packing capture:

- Online matching problems (e.g., advertising)
- Scheduling of multi-part tasks on machines
- Forms of the winner determination problem for some combinatorial auctions
- Barter exchanges and organ allocation

Theory is still developing for complex online problems

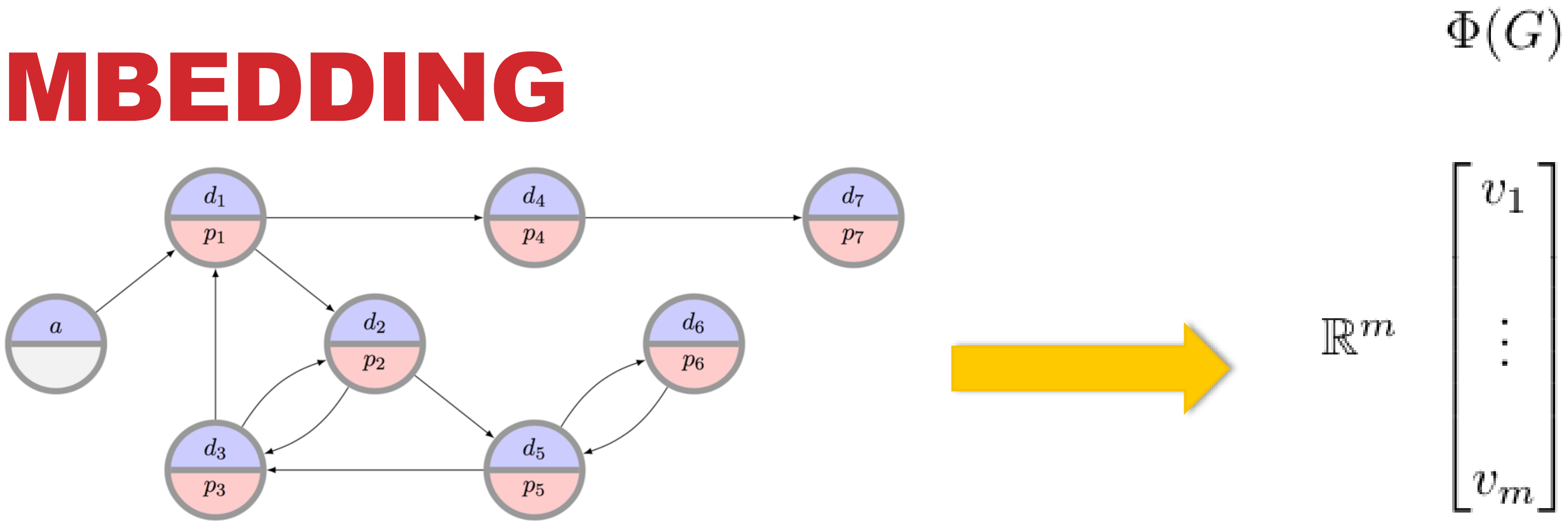
Can we learn complex, state-dependent matching and allocation policies in general environments?

LEARNING TO PACK & MATCH



1. **Embed** current state space (compatibility/feasibility graph) into fixed-dimensional space
2. **Neural network** takes vectors as input (use RL to learn appropriate policy)
3. Take an action (e.g., in simplest form, flip a **biased coin**)
4. If heads: find and **match** maximum cardinality matching
5. **Simulate** matching or allocation environment and update the current state

1. EMBEDDING

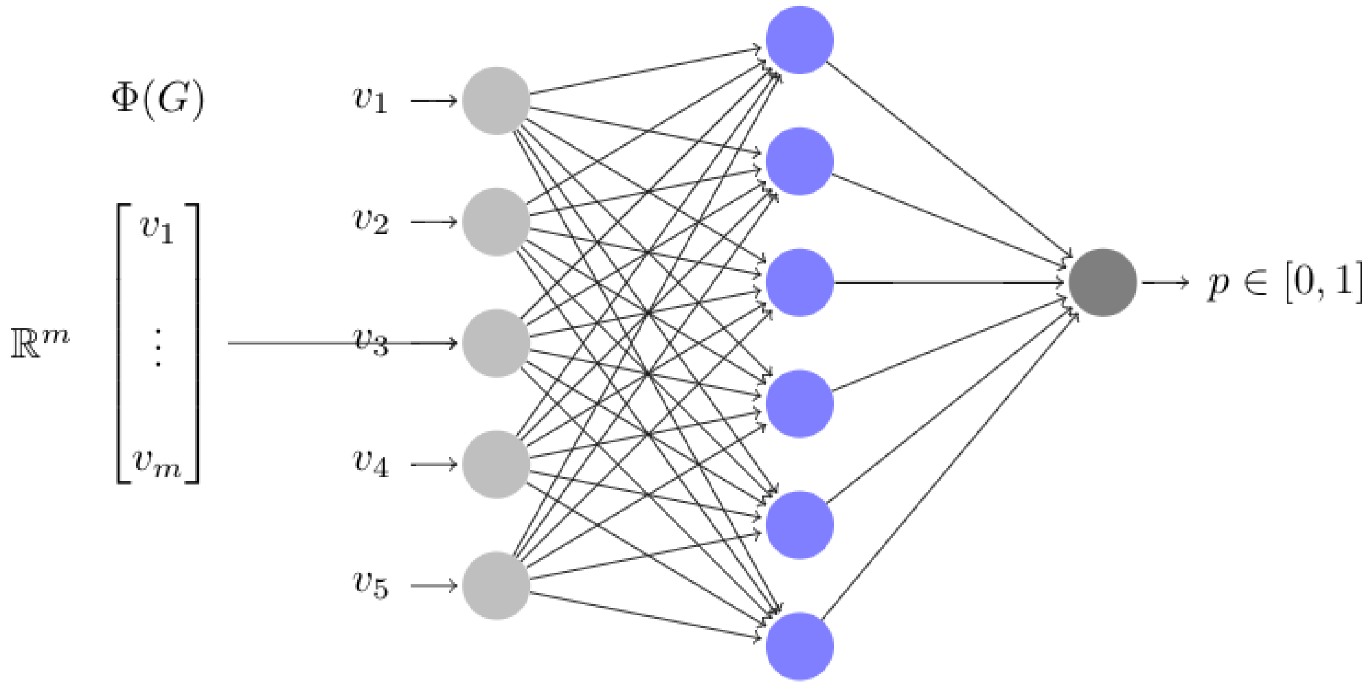


Need: Embed state (e.g., a graph) as a vector and still maintain certain properties, such as node neighborhood structure. We use random walks to do so [Li, Campbell, Caceres 2017]

Use random walk on a carefully selected initial distribution to capture temporal changes in probability distribution

- Encode distance between pairs of probability distributions
- Sanity check: can distinguish between families of random graphs (e.g., Erdős-Rényi and Stochastic Block Model), and real kidney exchange graphs

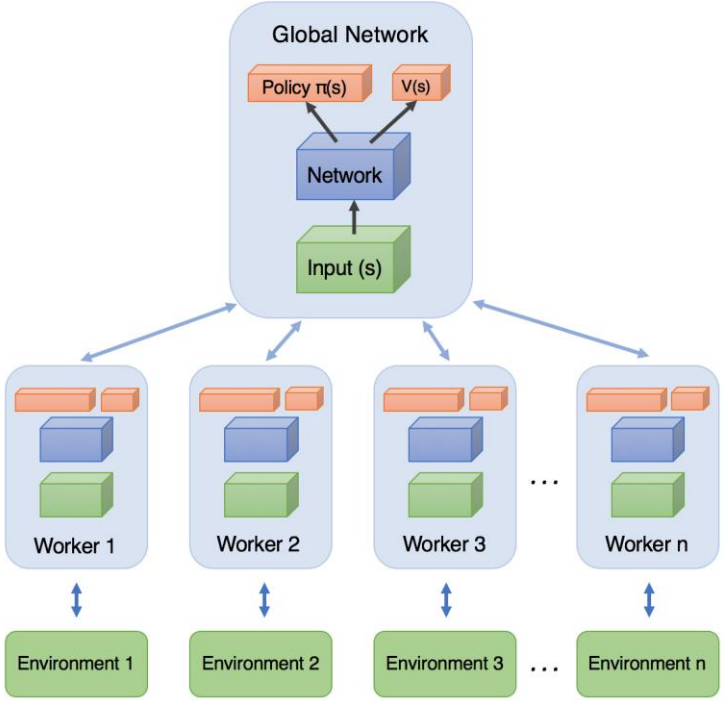
2. EMBEDDING TO NEURAL NET

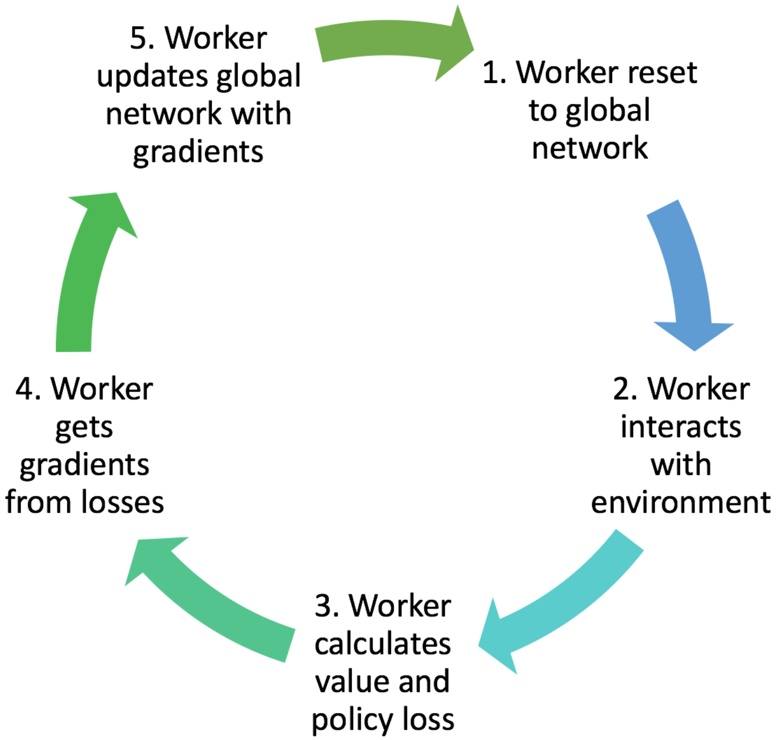


Feed an embedded graph into, e.g., a neural network to output a learned probability for our biased coin flip

- (Network structure and action space are much more complicated in practice)

3. LEARNING ALGORITHM





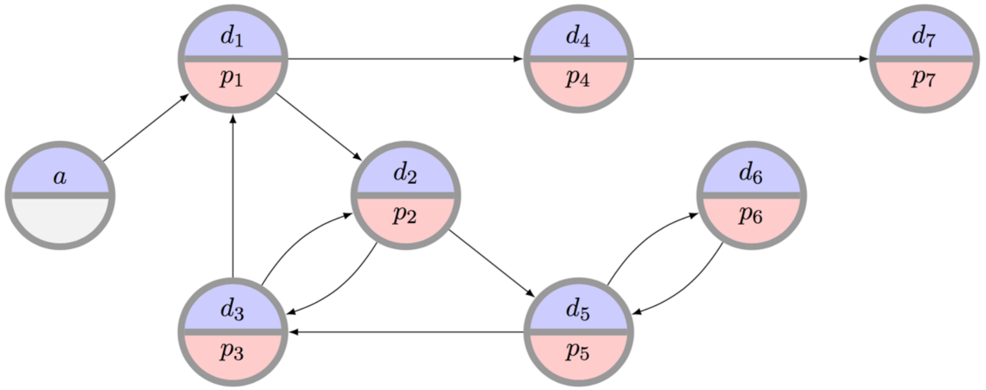
Using an adaptation of Asynchronous Advantage Actor-Critic (A3C) method [Mnih 2016]

4. MAKE A (PROBABILISTIC) MATCHING DECISION

(Simplest possible action space, for exposition!)

1

0



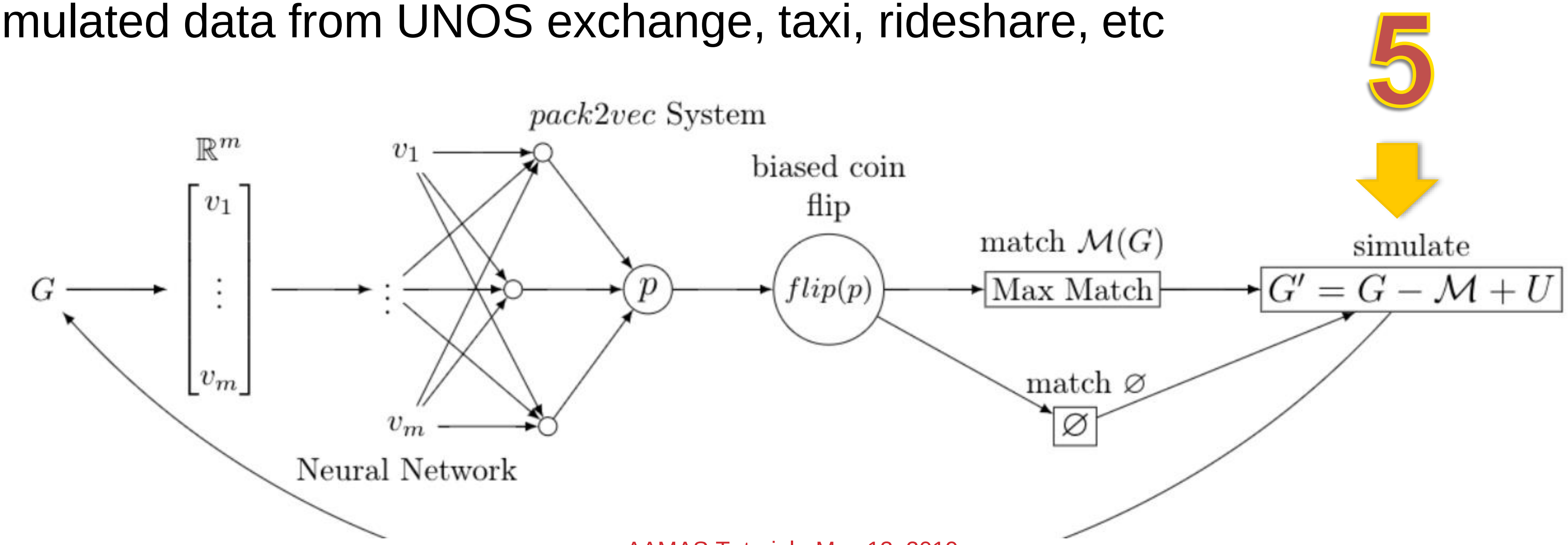
MAX MATCH

MATCH NOTHING (wait)

5. CLOSING THE LOOP FOR TRAINING

To train a matching network (as well as the embedding network), we must be able to simulate realistic environments

- Homogeneous Erdős–Rényi graphs [Akbarpour et al. 2017]
- Heterogeneous Erdős–Rényi graphs [Ashlagi et al. 2013]
- Real/Simulated data from UNOS exchange, taxi, rideshare, etc



EARLY RESULTS

Possible to replicate results from prior theory papers:

- In some models, dynamic matching helps
- In some models, dynamic matching does not help

Still iterating on:

- Neural net structure
- Action space (binary coin flip vs. multiple match types)
- Learning method (A3C vs. DQN vs. more standard methods)

But …

- **Seems promising.** Can learn matching policies beyond simply batching for T time periods; can realize gains over greedy.
- Policies depend on graph structure.

CASE STUDY: ONLINE DIVERSE BIPARTITE MATCHING

INTER-AGENT EFFECTS

(Weighted) matching market literature focuses on maximizing the sum of the utility of individual matches (subject to constraints).

- Not always the right idea!

Say you are a firm hiring workers: what is your goal?

Maximize the number of open positions filled …

… with "good" candidates …

… subject to fairness constraint(s) …

… and such that the entire hired cohort works well together!

EXTENSION TO THE ONLINE CASE

What if workers do not apply in a single batch?

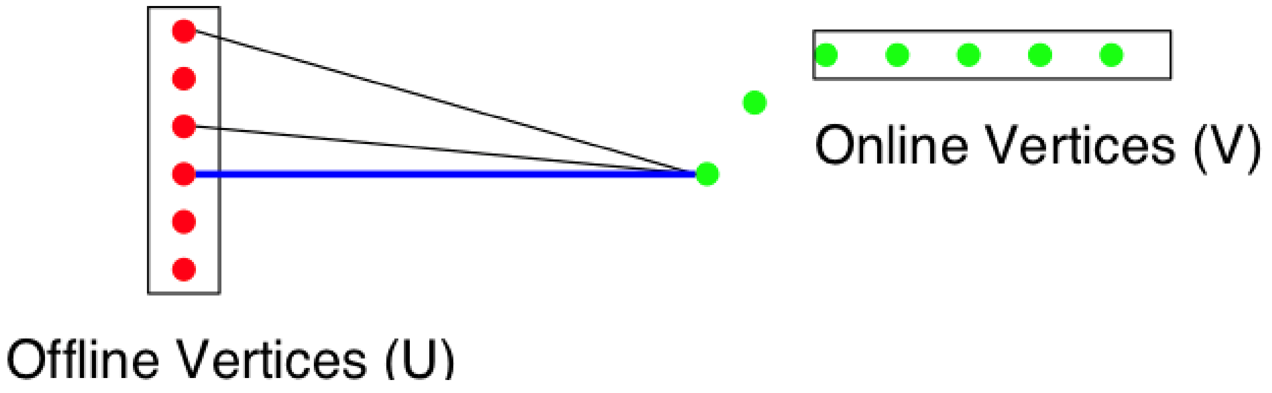
- E.g., matching workers to tasks in an online labor market
- May have soft constraints (synergies in the workforce)
- May have hard constraints (quota systems)

Another example: Internet advertising

- Reach: the number of individuals
- Frequency: the rate at which you select an individual
- Law of diminishing returns

Reach	Frequency	Impressions
12,586 People	1.29 Per Person	16,190 Total

ONLINE MODEL



Bipartite graph G = (U,V,E)

- Know U entirely offline
- V arrives one by one

T time steps

- At each time step *t*, sample *v* independently from known distribution *D*
- Vertex *v* must be assigned immediately and irrevocably, or rejected
- (Assume T >> 1, and |V| >> |U|)

Non-negative, monotone, submodular function *f* over the edges *E*

- Goal: design an algorithm ALG that finds matching M that maximizes **E**[*f*(M)]

DIVERSITY VIA SUBMODULARITY

Ground set of elements [*n*] := {1, 2, …, *n*}

$$f : 2^{[n]} \to \mathbb{R}^+$$

Function *f* is submodular if for every subset of elements *A* and *B*:

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$$

Function *f* is monotone if:

$$f(A) \leq f(B) \quad \forall A \subseteq B \subseteq [n]$$

TWO-PHASE APPROACH

Separate the algorithms into two phases:

- **Offline: obtain an upper bound on the offline optimal solution**
- **Online: use that to guide the online matching algorithm**

Multilinear extension of a submodular function *f* is

$$F(x) := \sum_{T \subseteq [n]} \left(\prod_{j \in T} x_j \prod_{j \notin T} (1 - x_j) \right) f(T)$$

High-level notes:

- $F(\mathbf{x}) = f(\mathbf{x})$ on any integral points
- If $\mathcal{R}_{\mathbf{x}} \subseteq [n]$ with elements packed according to distribution **x**, then $F(\mathbf{x}) = \mathbb{E}\left[f(\mathcal{R}_{\mathbf{x}})\right]$

OFFLINE PHASE

Solve the following program:

Expected matches for any *v* is at most the expected number of arrivals r_v

$$\begin{aligned} \text{maximize} \quad & F(\mathbf{x}) \\ \text{subject to} \ & \textstyle\sum_{e \in E(v)} x_e \le r_v & \forall v \in V \\ & \textstyle\sum_{e \in E(u)} x_e \le 1 & \forall u \in U \\ & 0 \le x_e \le 1 & \forall e \in E \end{aligned}$$

Expected number of matches for any *u* is at most 1

If *f* is linear, can solve this exactly

If *f* is monotone submodular, solve approximately **[Calinescu et al. 2011, Adamczyk et al. 2016]**

Yields probability distribution x* from which we can sample during online phase

ONLINE PHASE: TWO ALGORITHMS

First, solve previous program for $x^* = (x^*_e)$, such that $F(x^*) \geq (1 - 1/e)$ OPT …

Multilinear Maximization Program (MMP-ALG):

- v arrives: sample edge $e = (u,v)$ from its neighbors with probability x^*_e / r_v
- Match if u is safe; else reject

Contention Resolution (CR-ALG):

- Uses techniques from contention resolution (CR) literature [Vondrak et al. 2011]
- Roughly, sample a binary vector **Y** in the offline phase based on **x***
- Match according to **Y** in the online phase if safe; else reject

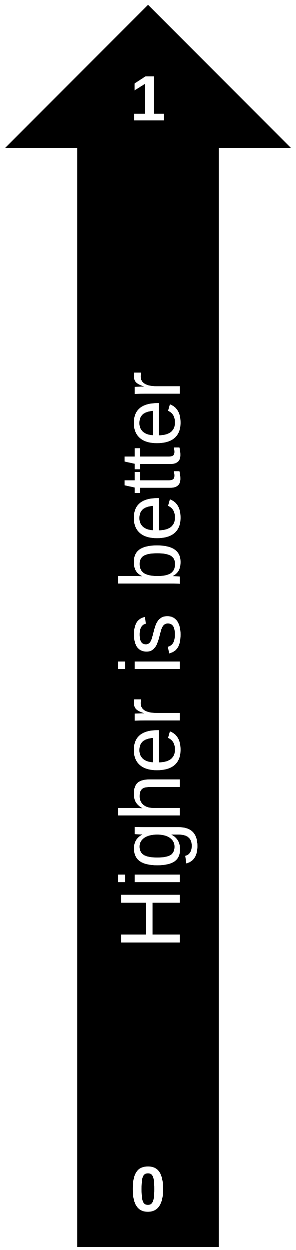
COMPETITIVE RATIO

In: An instance of the problem
X_{ALG}(In): Random variable to denote weight of matching in ALG
X_{OPT}(In): Random variable to denote weight of matching in OPT

$$\text{Competitive Ratio} = \min_{In} E[X_{ALG}(In)] \,/\, E[X_{OPT}(In)]$$

E[numerator]: over internal randomness of algorithm and arrival sequence

E[denominator]: over randomness in arrival sequence



THEORETICAL RESULTS

CR-ALG achieves an online competitive ratio of at least

$$\frac{1}{2}(1 - e^{-1/2})(1 - 1/e)$$

when arrival rates are integral

MMP-ALG achieves an online competitive ratio of at least

$$(1 - 1/e)^2$$

when $|U| = o(\sqrt{T})$

EARLY, EXPLORATORY EXPERIMENTS

Run experiments on two common submodular functions

Budget additive:

- Maximize the sum of weights subject to a budget constraint *B*

$$\max_{M \in \mathcal{M}} f(M) = \min \left\{ B, \sum_{e \in M} w_e \right\}$$

Weighted coverage of a set of elements

Framework:

- Extension to *b*-matching (only upper bounds) on the fixed U side, varies "constrainedness"
- Benchmark: offline optimal (solve a big, omniscient math program to optimality)

EARLY EXPERIMENTS

NEG-CR: small tweak where uniform sampling is replaced by dependent rounding [Gandhi et al 2006]

Learn semi-matching offline, sample from it online

Greedy: choose an available neighbor that maximizes marginal gain in *f*, otherwise drop *v*

MMP-ALG and CR-ALG from earlier

Early results:

- Budget: much better than Greedy, much better than c-r
- Coverage: Greedy wins for constrained *b* → maybe better algorithms than MMP-ALG and CR-ALG?