APPLIED MECHANISM DESIGN FOR SOCIAL GOOD

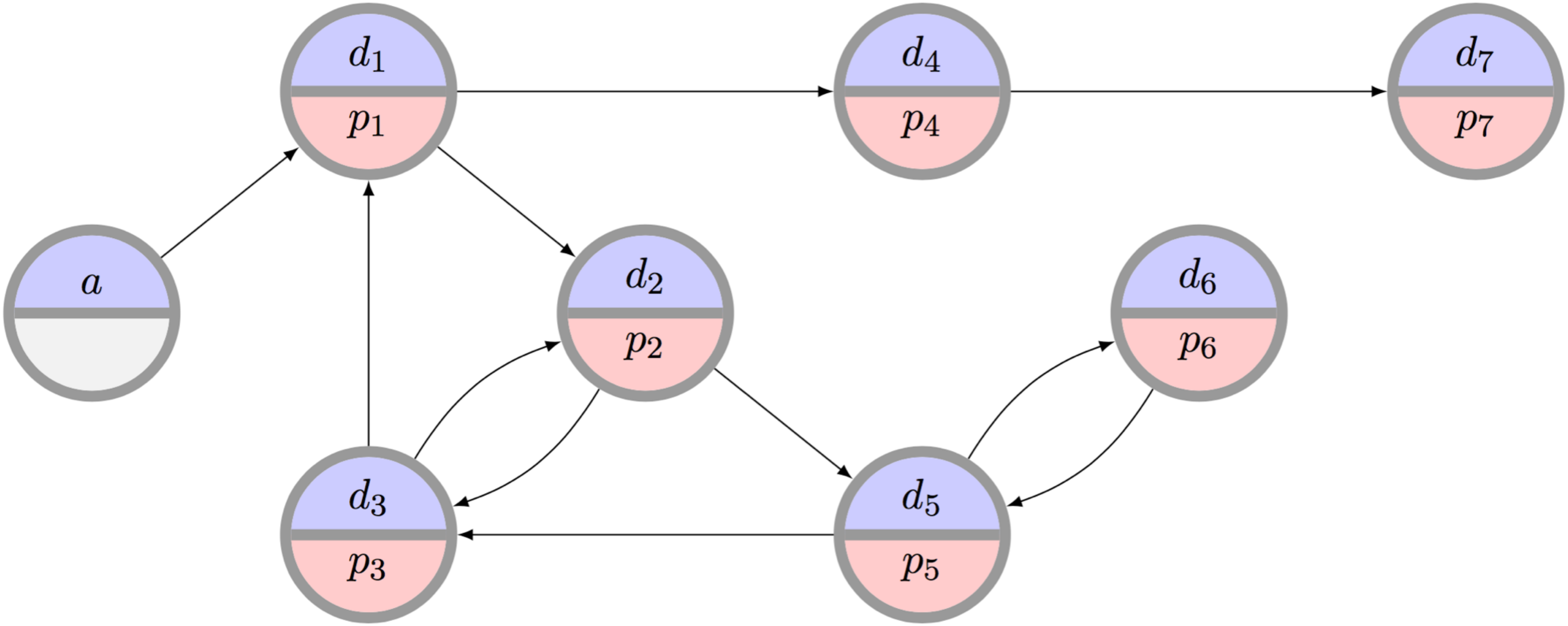
JOHN P DICKERSON

Lecture #19 – 04/14/2020

CMSC828M
Tuesdays & Thursdays
2:00pm – 3:15pm

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

THE CLEARING PROBLEM

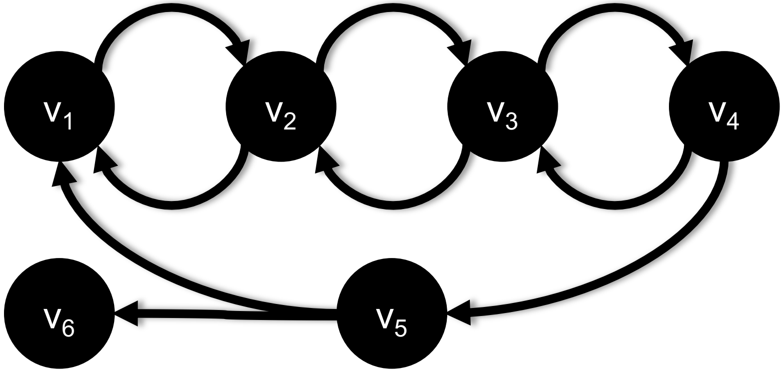


The clearing problem is to find the "best" disjoint set of cycles of length at most *L*, and chains

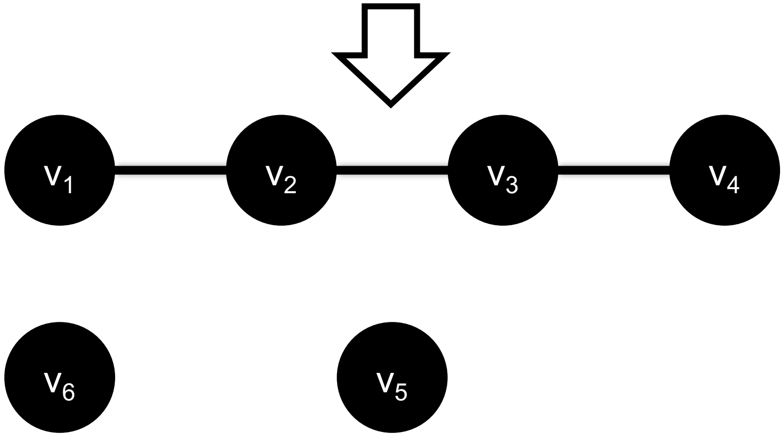
- Typically, 2 ≤ *L* ≤ 5 for kidneys (e.g., *L*=3 at UNOS)
- NP-hard (for *L*>2) in theory, really hard in practice [Glorie et al. 2014, Anderson et al. 2015, Plaut et al. 2016, Dickerson et al. 2016 ...]
  [Abraham et al. 07, Biro et al. 09]

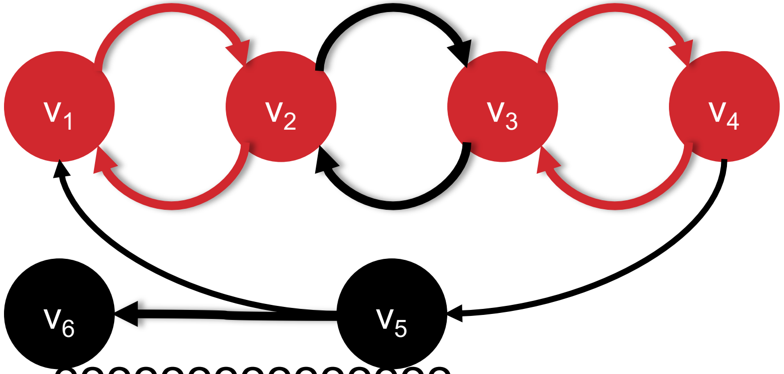
SPECIAL CASE: *L* = 2

PTIME: translate to maximum matching on undirected graph

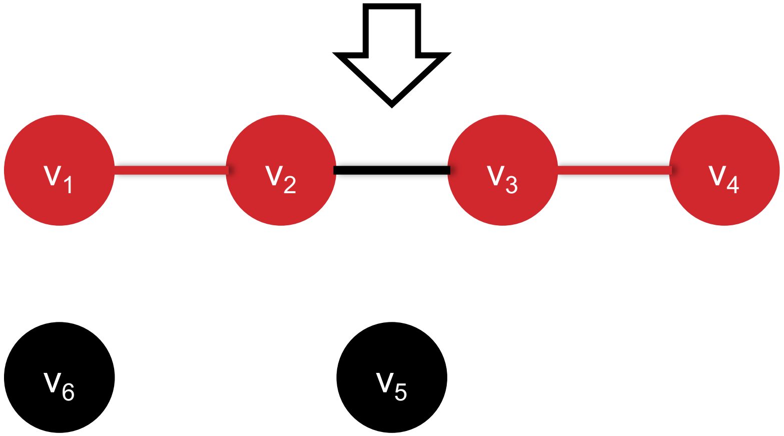


(Six pairs, no altruists.)



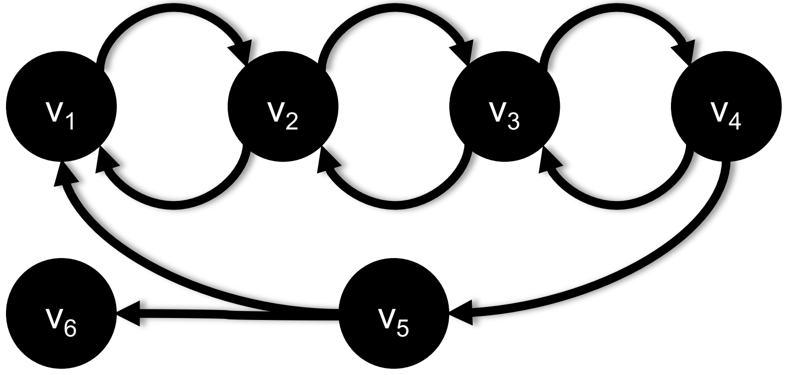


???????????????

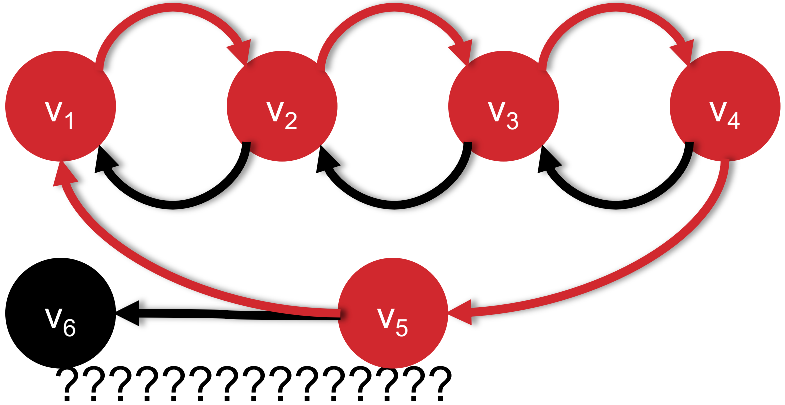


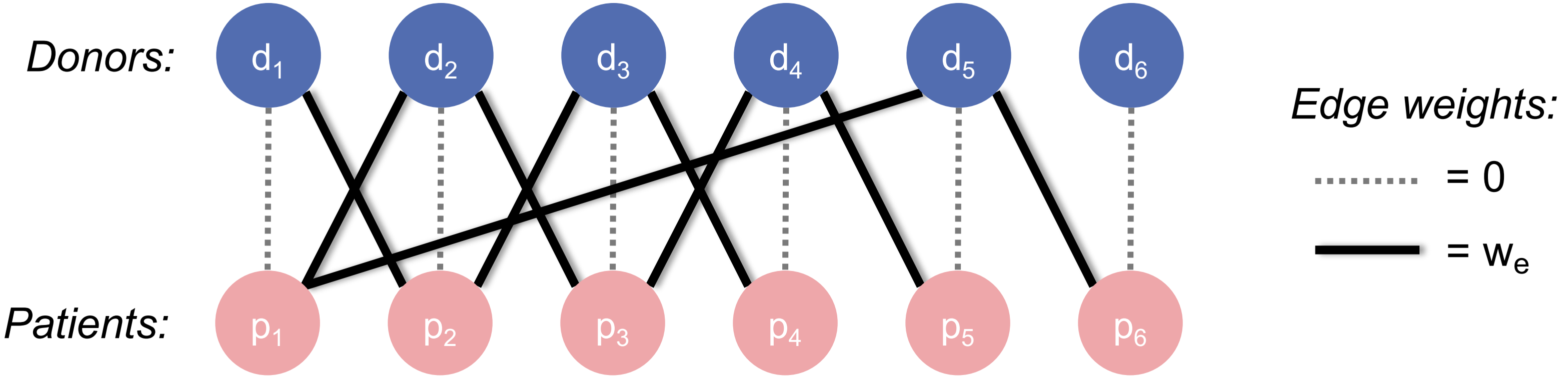
SPECIAL CASE: *L* = ∞

PTIME via formulation as maximum weight perfect matching



(Six pairs, no altruists.)

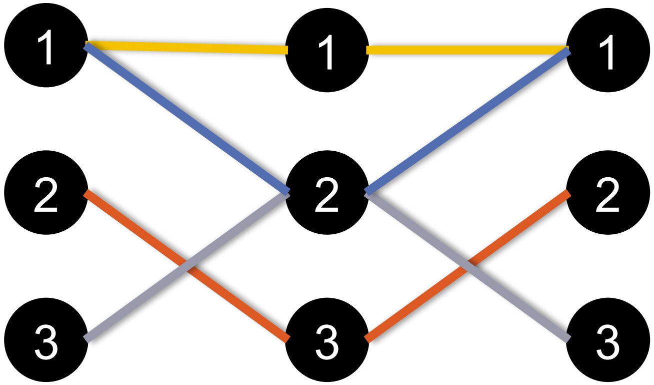




GENERAL CASE: *L* = ?

NP-hard via reduction from 3D-matching:

- **Given disjoint sets X, Y, Z of size *q* …**
- **... and a set of triples T ⊆ X x Y x Z ...**
- **... is there a disjoint subset M ⊆ T of size *q*?**



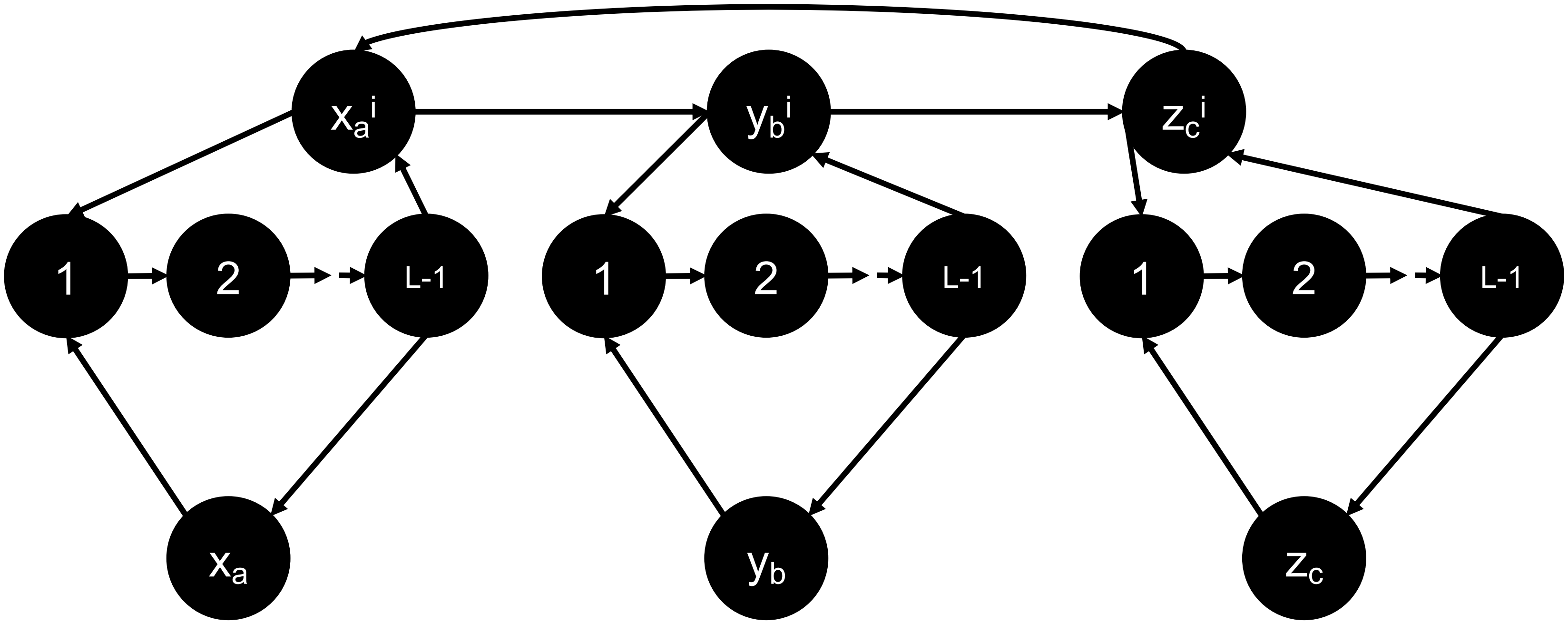
T = {
(1,1,1), ✓
(2,3,2), ✓
(1,2,1),
(3,2,3), ✓
}

???????????????

GENERAL CASE: *L* = ?

Construct a gadget for each $t_i = \{x_a, y_b, z_c\}$ in T

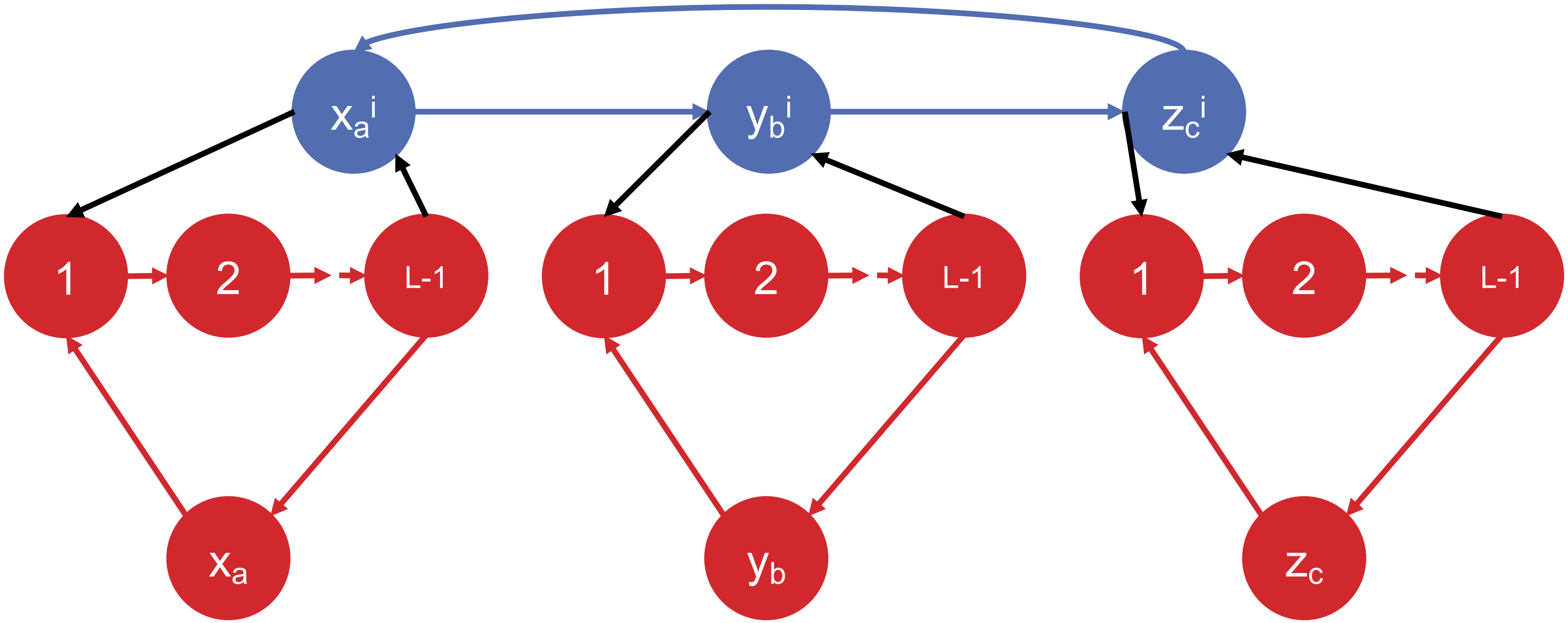
- Gadgets intersect only on vertices in X U Y U Z



GENERAL CASE: *L* = ?

M is perfect matching → construction has perfect cycle cover.

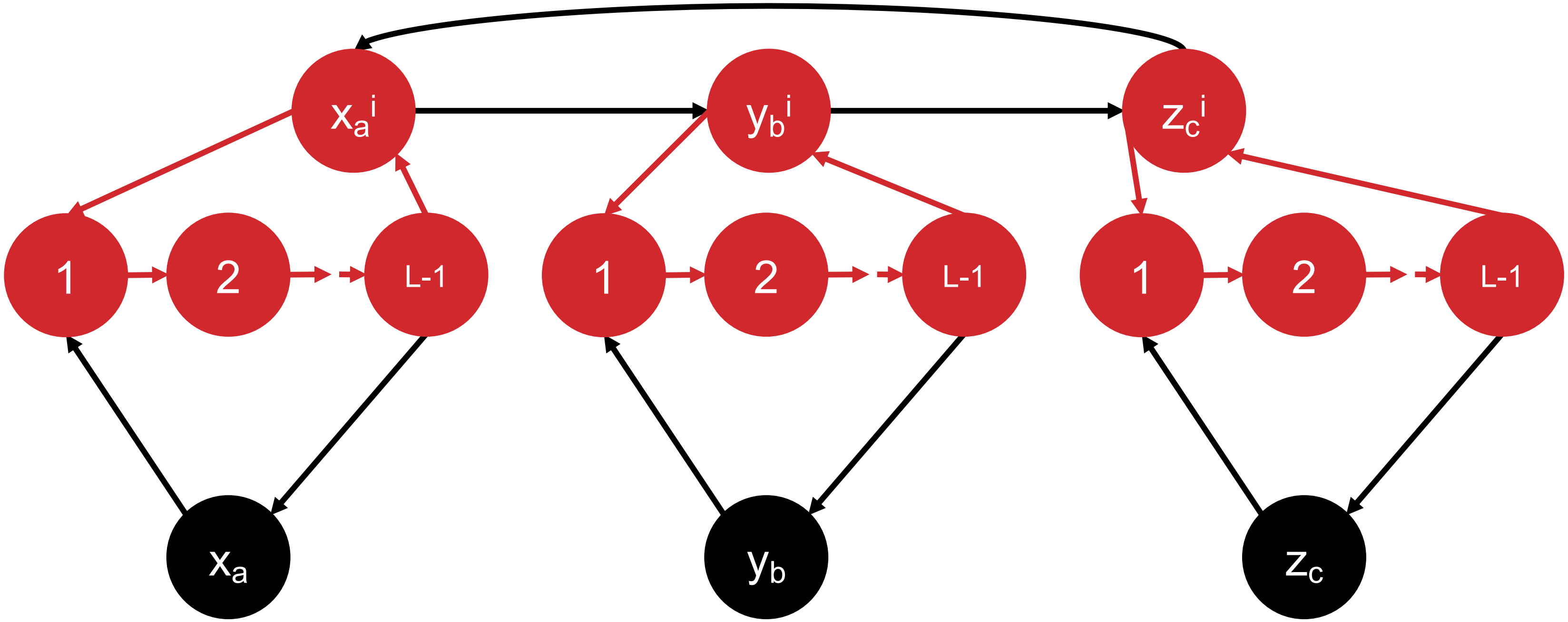
For t_i in *T*:



GENERAL CASE: *L* = ?

M is perfect matching → construction has perfect cycle cover.

For t_i not in *T*:



GENERAL CASE: *L* = ?

We have a perfect cycle cover → *M* is a perfect 3D matching

- Construction only has 3-cycles and *L*-cycles
- Short cycles (i.e., 3-cycles) are disjoint from the rest of the graph by construction

Thus, given a perfect cover (by assumption):

- Widgets either contribute according to t_i in M …
- … or t_i not in M.

Thus there is a perfect matching in the original 3D matching instance.

HOPELESS ...?

BASIC APPROACH #1: THE EDGE FORMULATION

[Abraham et al. 2007]

Binary variable x_{ij} for each edge from *i* to *j*

Maximize

$$u(M) = \Sigma\, \mathrm{w}_{\mathrm{ij}}\, x_{ij}$$

Flow constraint

Subject to

$\Sigma_{\mathrm{j}}\, x_{ij} = \Sigma_{\mathrm{j}}\, x_{ji}$ for each vertex *i*

$\Sigma_{\mathrm{j}}\, x_{ij} \leq 1$ for each vertex *i*

$\Sigma_{1 \leq \mathrm{k} \leq \mathrm{L}}\, \mathrm{x}_{\mathrm{i(k)i(k+1)}} \leq \mathrm{L}\text{-}1$ for paths i(1)…i(L+1)

(*no path of length L that doesn't end where it started – cycle cap)*

STATE OF THE ART FOR EDGE FORMULATION

[Anderson et al. PNAS-2015]

Builds on the prize-collecting traveling salesperson problem [Balas Networks-89]

- PC-TSP: visit each city (patient-donor pair) exactly once, but with the additional option to pay some penalty to skip a city (penalized for leaving pairs unmatched)

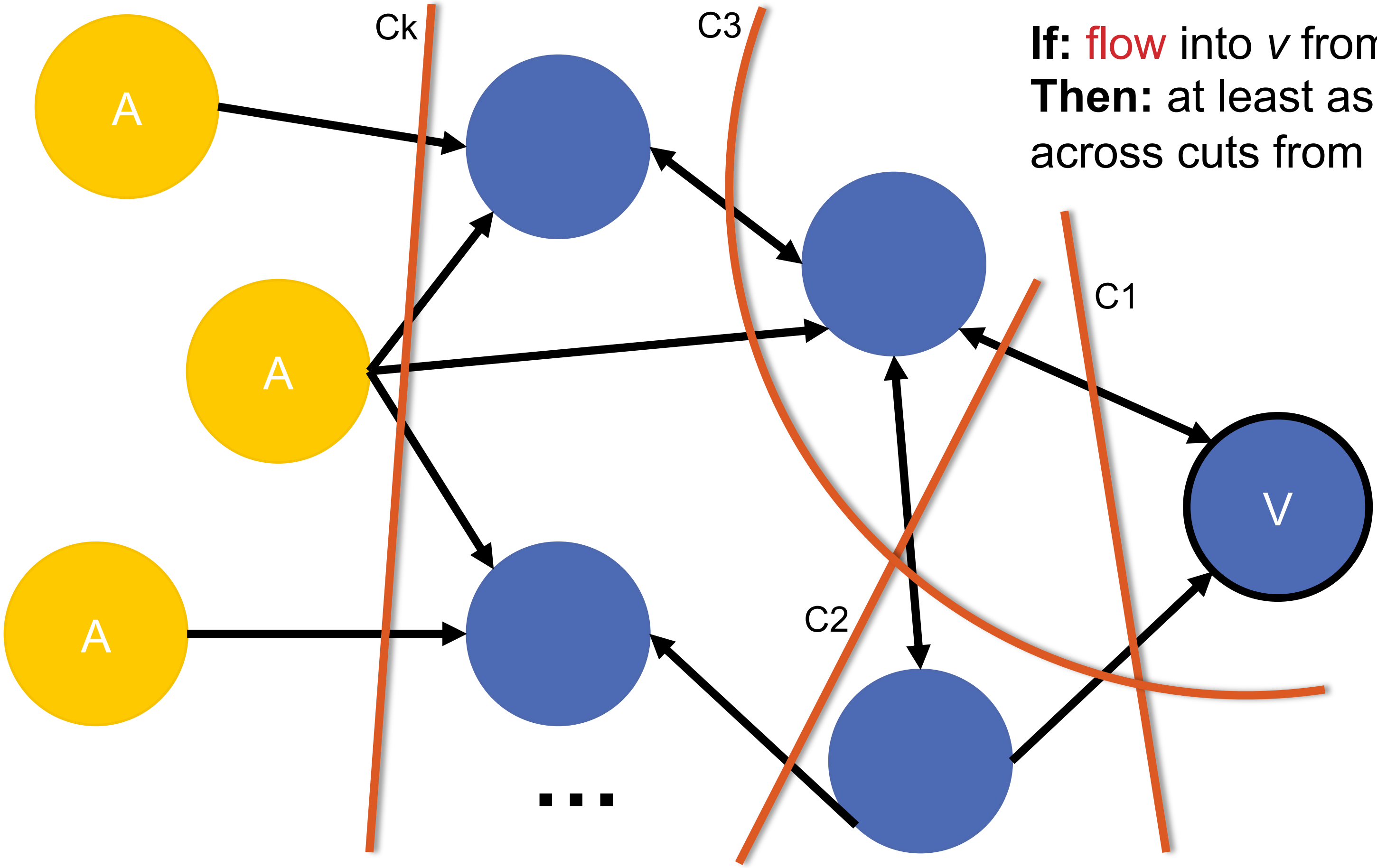
They maintain decision variables for all cycles of length at most *L*, but build chains in the final solution from decision variables associated with individual edges

Then, an exponential number of constraints could be required to prevent the solver from including chains of length greater than *K*; these are generated incrementally until optimality is proved.

- Leverage cut generation from PC-TSP literature to provide stronger (i.e. tighter) IP formulation

BEST EDGE FORMULATION

[Anderson et al. 2015]



If: flow into *v* from a chain
Then: at least as much flow across cuts from {A}

BASIC APPROACH #2: THE CYCLE FORMULATION

[Roth et al. 2004, 2005,
Abraham et al. 2007]

Binary variable x_c for each feasible cycle or chain c

Maximize

$$u(M) = \Sigma\, w_c\, x_c$$

Subject to

$$\Sigma_{c\,:\,i \text{ in } c}\, x_c \leq 1 \text{ for each vertex } i$$

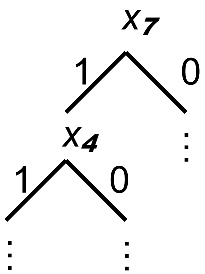
SOLVING THE CYCLE FORMULATION IP

Too large to write down

- $O(\max\{\, |P|^L, |A||P|^{K-1} \})$ variables
- $|A| = 5$, $|V|=300$, $L=3$, $K=20$ … $|A||P|^{K-1} \approx 5 \times 10^{47}$

Approach: branch-and-price [Barnhart et al. 1998]:

- Branch: select fractional column and fix its value to 1 and 0 respectively



- Fathom the search node if no better than incumbent
  - Solve LP relaxation using column generation

COLUMN GENERATION

Master LP *P* has too many variables

- Won’t fit in memory, and/or would take too long to solve

Begin with restricted LP *P*’, which contains only a small subset of the variables (i.e., cycles)

- OPT(*P*’) ≤ OPT(*P*)

Solve *P*’ and, if necessary, add more variables to it

- We do this intelligently by solving the pricing problem

Repeat until OPT(*P*’) = OPT(*P*)

DFS TO SOLVE PRICING PROBLEM

[Abraham et al. EC-07]

Pricing problem:

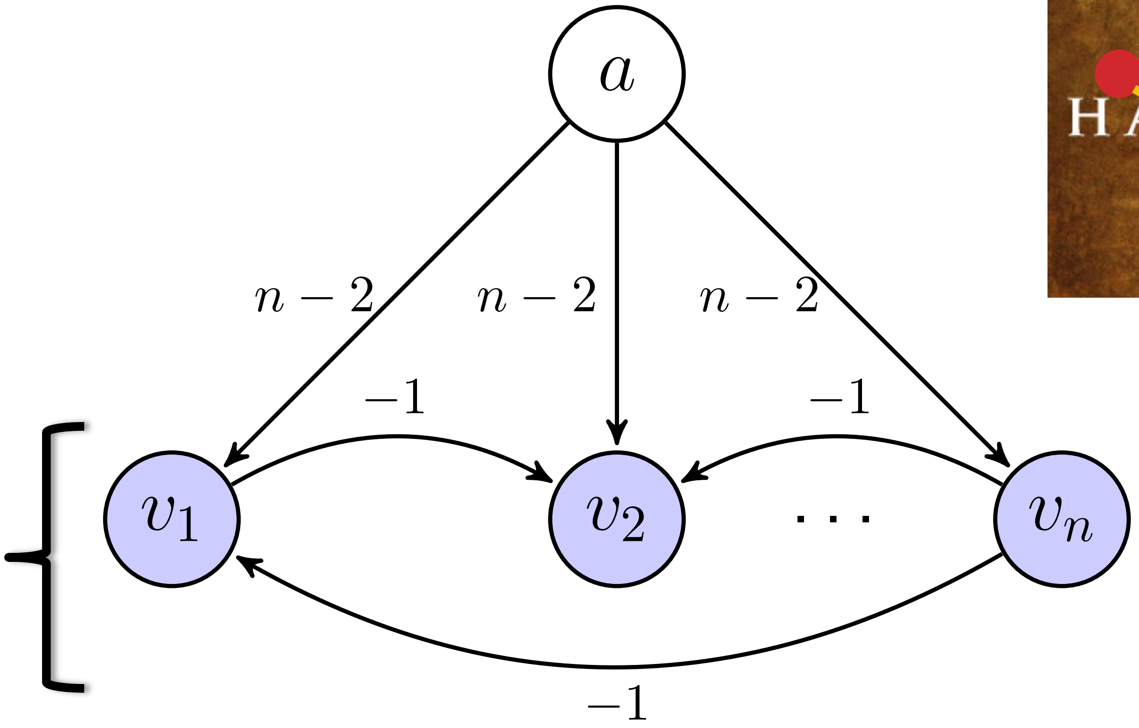
- Optimal dual solution π^* to reduced model
- Find non-basic variables with **positive price** (for a maximization problem)
  - 0 < weight of cycle – sum of duals in π^* of constituent vertices
  - Positive price for cycle → dual constraint is violated
  - No positive price cycles → no dual constraints violated

First approach [Abraham et al. EC-2007] explicitly prices all feasible cycles and chains through a DFS

- Can speed this up in various ways, but proving **no positive price cycles exist** still takes a long time

GENERAL PRICING OF CYCLES & CHAINS IS NP-HARD [Plaut et al. arXiv:1606.00117]

Reduce from Hamiltonian path



Arbitrary graph G

COMPARISON

Tradeoffs in number of variables, constraints

- IP #1: $O(|E|^L)$ constraints vs. $O(|V|)$ for IP #2
- IP #1: $O(|V|^2)$ variables vs. $O(|V|^L)$ for IP #2

IP #2's relaxation is weakly tighter than #1's. Quick intuition in one direction:

- Take a length L+1 cycle. #2's LP relaxation is 0.
- #1's LP relaxation is $(L+1)/2$ – with ½ on each edge

Recent work focuses on balancing tight LP relaxations and model size [Constantino et al. 2013, Glorie et al. 2014, Klimentova et al. 2014, Alvelos et al. 2015, Anderson et al. 2015, Mak-Hau 2015, Manlove&O'Malley 2015, Plaut et al. 2016, …]**:**

- Newest work: compact formulations, some with tightest relaxations known, all amenable to failure-aware matching

COMPACT FORMULATIONS [Constantino et al. EJOR-14]

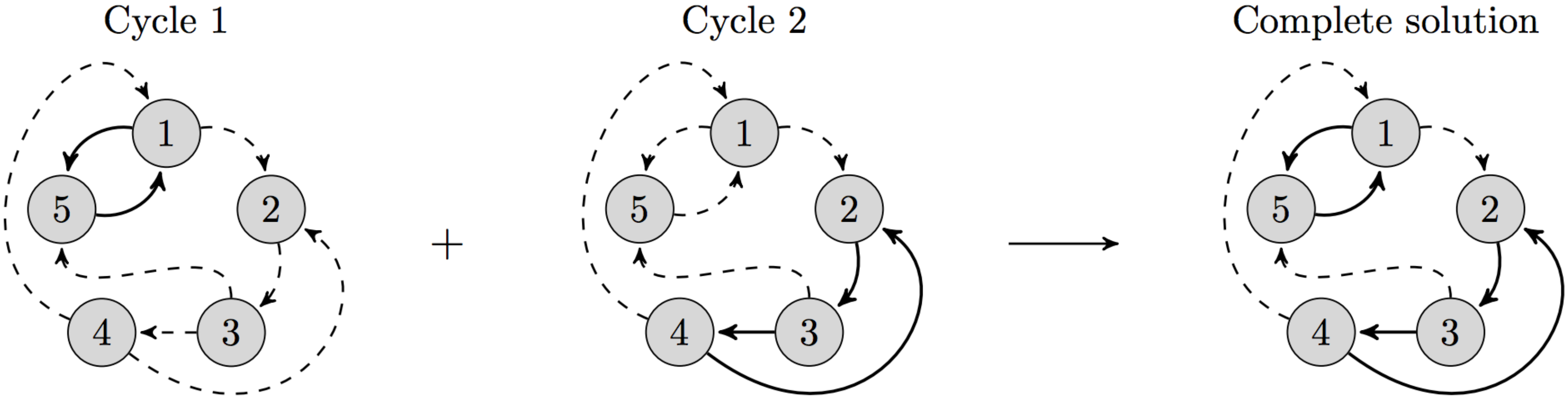
Previous models: exponential #constraints (CG methods) or #variables (B&P methods)

Let F be upper bound on #cycles in a final matching

Create F copies of compatibility graph

Search for a single cycle or chain in each copy

- (Keep cycles/chains disjoint across graphs)



COMPACT FORMULATIONS

$$x_{ij}^f = \begin{cases} 1 & \text{if arc } (i,j) \text{ is selected to be in copy } f \text{ of the graph,} \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{array}{lll} \text{maximize} & \sum_f \sum_{(i,j)\in A} w_{ij} x_{ij}^f & & \text{1A} \\ \text{subject to} & \sum_{j:(j,i)\in A} x_{ij}^f = \sum_{j:(i,j)\in A} x_{ij}^f & \forall i \in V, \forall f \in \{1,\ldots,F\} & \text{1B} \\ & \sum_f \sum_{j:(i,j)\in A} x_{ij}^f \leq 1 & \forall i \in V & \text{1C} \\ & \sum_{(i,j)\in A} x_{ij}^f \leq k & \forall f \in \{1,\ldots,F\} & \text{1D} \\ & x_{ij}^f \in \{0,1\} & \forall (i,j) \in A, \forall f \in \{1,\ldots,F\} & \text{1E} \end{array}$$

1A: max edge weights over all graph copies

1B: give a kidney <-> get a kidney within that copy

1C: only use a vertex once

1D: cycle cap

Polynomial #constraints and #variables!

PIEF: A COMPACT MODEL FOR CYCLES ONLY

[Dickerson Manlove Plaut Sandholm Trimble EC-16]

Builds on Extended Edge Formulation of Constantino et al.

- **O(|*V*|) copies of graph, 1 binary variable per edge per copy**
- **Enforce at most one cycle per graph copy used**
- **Track positions of edges in cycles for LP tightness**

THEOREM

The tightest known non-compact LP relaxation

$Z_{CF} = Z_{PIEF}$

(disallowing chains)

(EC-16 paper also presents HPIEF, which is a compact formulation for cycles and chains, but with weaker Z_{HPIEF})

PICEF: POSITION-INDEXED CHAIN-EDGE FORMULATION

In practice, cycle cap *L* is small and chain cap *K* is large

Idea: enumerate all cycles but not all chains [Anderson et al. 2015]

- That work required $O(|V|^K)$ constraints in the worst case
- This work requires $O(K|V|) = O(|V|^2)$ constraints

MAIN IDEA

Track not just if an edge is used in a chain, but where in a chain an edge is used.

For edge (i,j) in graph: $K'(i,j) = \{1\}$ if *i* is an altruist

$K'(i,j) = \{2, \ldots, K\}$ if *i* is a pair

PICEF: POSITION-INDEXED CHAIN-EDGE FORMULATION

Maximize

$$u(M) = \Sigma_{ij \text{ in } E} \Sigma_{k \text{ in } K'(i,j)} w_{ij} y_{ijk} + \Sigma_{c \text{ in } C} w_c z_c$$

Subject to

$\Sigma_{ij \text{ in } E} \Sigma_{k \text{ in } K'(i,j)} y_{ijk} + \Sigma_{c : i \text{ in } c} z_c \leq 1$ for every *i* in *Pairs*

Each pair can be in at most one chain or cycle

$\Sigma_{ij \text{ in } E} y_{ij1} \leq 1$ for every *i* in *Altruists*

Each altruist can trigger at most one chain via outgoing edge at position 1

$\Sigma_{j:ij \text{ in } E} y_{ijk+1} - \Sigma_{j:ji \text{ in } E \wedge k \text{ in } K'(j,i)} y_{jik} \leq 0$ for every *i* in *Pairs* and *k* in {1, …, *K*-1}

Each pair can be have an outgoing edge at position k+1 in a chain iff it has an incoming edge at position k in a chain

WHAT IF THERE ARE STILL TOO MANY VARIABLES?

In particularly dense graphs or if, in the future, longer cycle caps are allowed, PICEF may need too many cycle variables

Solve via branch and price by storing only a subset of columns in memory, then solving pricing problem

- Search for variables with positive price, bring into model
- Previously: that search is exponential in chain cap [Abraham et al. 2007, Glorie et al. 2014, Plaut et al. 2016]
- General: pricing chains & cycle is NP-hard [arXiv:1606.00117]

But we only need to price cycles, not chains!

PRICING

PICEF is the first branch-and-price-based model with provably correct polynomial-time pricing

POLYNOMIAL-TIME CYCLE PRICING

[Glorie et al. MSOM-2014, Plaut et al. AAAI-2016]

Solve a structured problem that implicitly prices variables

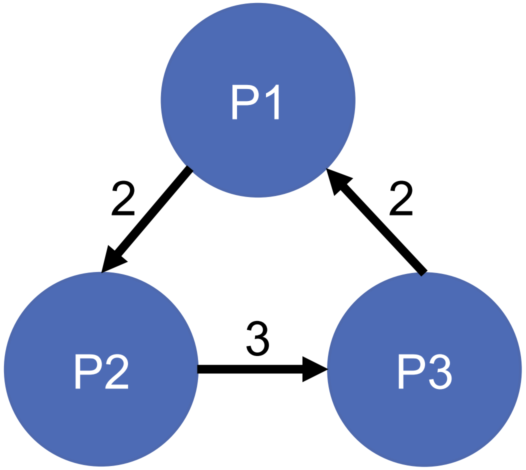
- Variable = x_c for cycle (not chain) c
- Price of x_c = $w_c - \Sigma_{v \text{ in } c}\, \delta_v$

Example

- Price: $\underbrace{(2+3+2)}_{w_c} - (\delta_{P1}+\delta_{P2}+\delta_{P3})$

$$= \Sigma_{e \text{ in } c}\, w_e - \Sigma_{v \text{ in } c}\, \delta_v$$

$$= \Sigma_{(u,v) \text{ in } c}\, [w_{(u,v)} - \delta_v]$$



Idea: Take *G*, create *G'* s.t. all edges *e* = (*u*,*v*) are reweighted $r_{(u,v)} = \delta_v - w_{(u,v)}$

- **Positive price cycles in *G* = negative weight cycles in *G'***

ADAPTED BELLMAN-FORD PRICING FOR CYCLES ONLY

[Glorie et al. MSOM-2014, Plaut et al. AAAI-2016]

Bellman-Ford finds shortest paths

- **Undefined in graphs with negative weight**
- **Adapt B-F to prevent internal looping during the traversal**
  - *Shortest* path is NP-hard (reduce from Hamiltonian path):
    - Set edge weights to -1, given edge (u,v) in E, ask if shortest path from u to v is weight 1-$|V|$ → visits each vertex exactly once
  - We only need *some* short path (or proof that no negative cycle exists)
- **Now pricing runs in time $O(|V||E|L^2)$**

HOW DO ALL THESE MODELS PERFORM IN PRACTICE?

Test on real and simulated match runs from:

- US UNOS exchange: 143+ transplant centers
- UK NLDKSS: 20 transplant centers

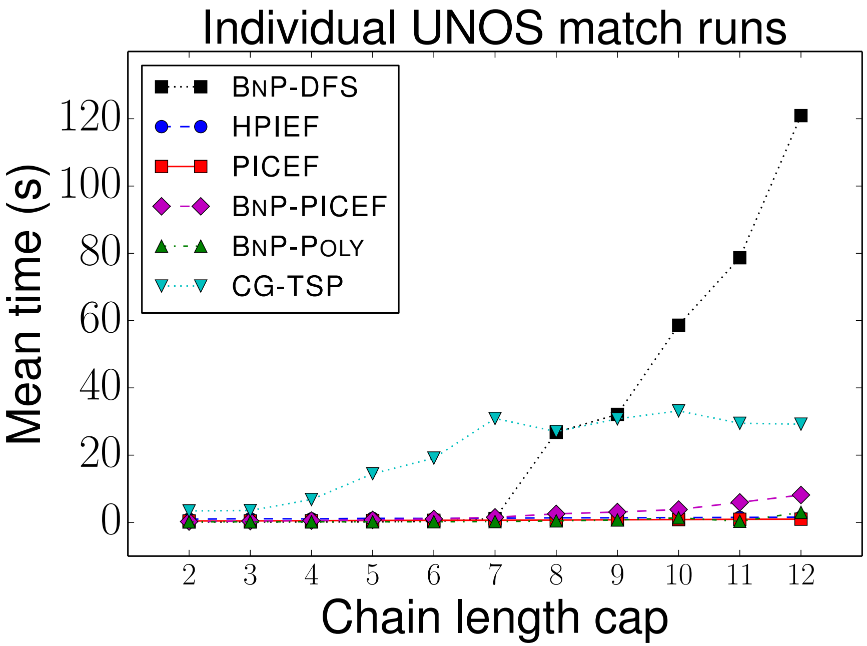
Following are tests against actual code for:

- BnP-DFS [Abraham et al. EC-07]
- BnP-Poly [Glorie et al. MSOM-14, Plaut et al. AAAI-16]
- CG-TSP [Anderson et al. PNAS-15]

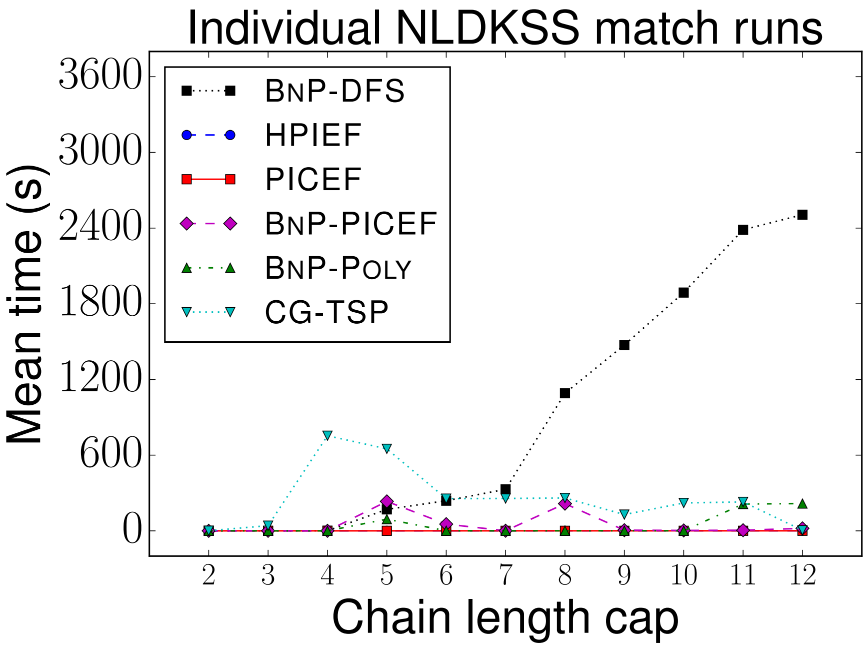
REAL MATCH RUNS

UNOS & NLDKSS

UNOS: 286 match runs

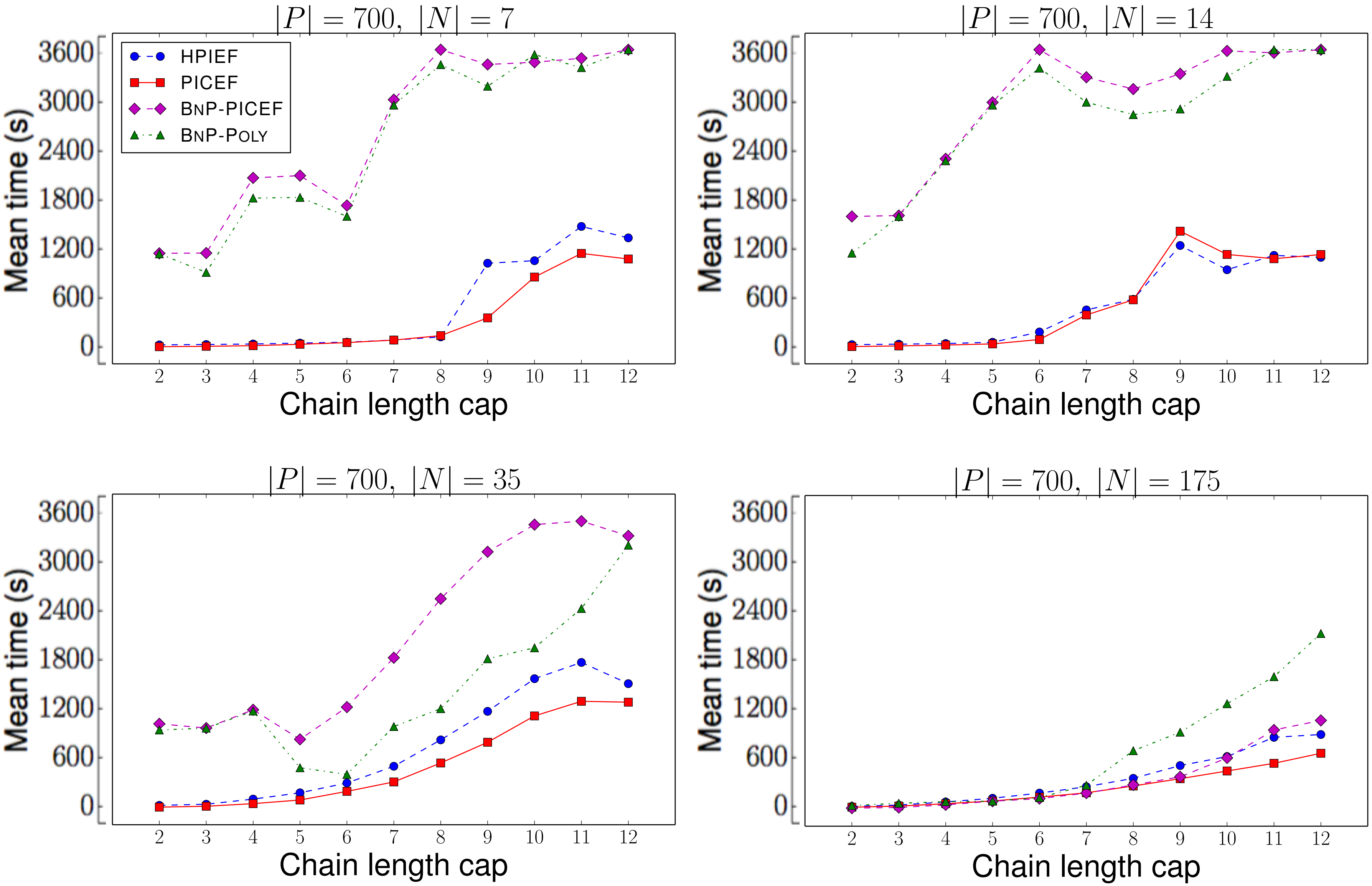


NLDKSS: 17 match runs



GENERATED DATA

$|P|$=700, INCREASING %ALTRUISTS



Solvers that are not shown timed out (within one-hour period).

THE BIG PROBLEM

What is “best”?

- Maximize matches right now or over time?
- Maximize transplants or matches?
- Prioritization schemes (i.e. fairness)?
- Modeling choices?
- Incentives? Ethics? Legality?

Optimization can handle this, but may be inflexible in hard-to-understand ways (for humans)

Want humans in the loop at a **high level**
(and then CS/Opt handles the implementation)

MANAGING SHORT-TERM UNCERTAINTY

[EC-13, EC-15, EC-16, Management Science *to appear*]

With A. Blum, N. Haghtalab, D. Manlove, B. Plaut, A. Procaccia, T. Sandholm, A. Sharma, J. Trimble

MATCHED ≠ TRANSPLANTED

Only around 10-15% of UNOS matched structures result in an actual transplant

- Similarly low % in other exchanges [ATC 2013]

***Many* reasons for this. How to handle?**

One way: encode *probability of transplantation* rather than just feasibility

- for individuals, cycles, chains, and full matchings

FAILURE-AWARE MODEL

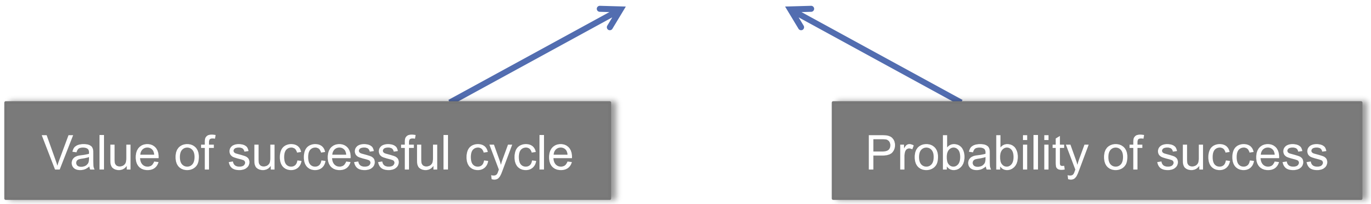
Compatibility graph *G*

- Edge (v_i, v_j) if v_i's donor can donate to v_j's patient
- Weight w_e on each edge *e*

Success probability q_e for each edge *e*

Discounted utility of cycle *c*

$$u(c) = \sum w_e \cdot \prod q_e$$

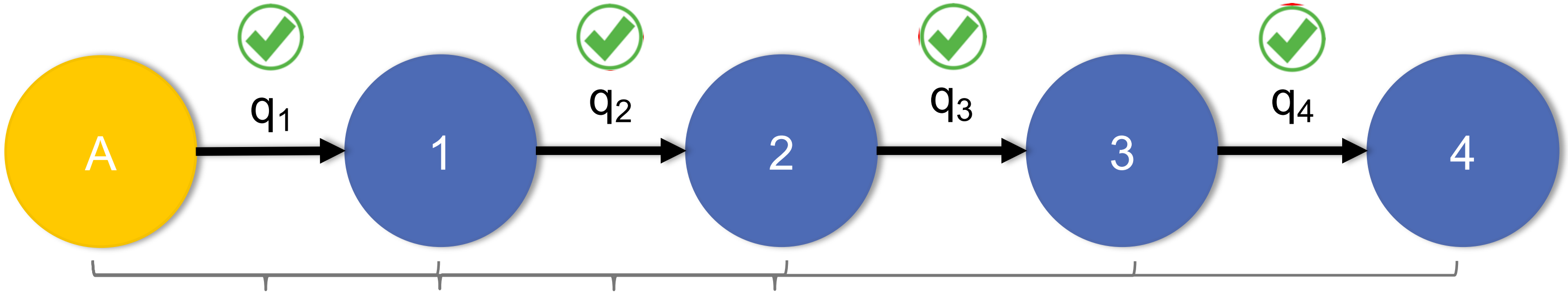


Value of successful cycle

Probability of success

FAILURE-AWARE MODEL

Discounted utility of a *k*-chain *c*



$1q_1(1-q_2). + 2q_1q_2(1-q_3)3q_1q_2q_3(1-q_4)4q_1q_2q_3q_4$

$$u(c) = \left[\sum_{i=1}^{k-1}(1-q_i)i\prod_{j=0}^{i-1}q_j\right] + \left[k\prod_{i=0}^{k-1}q_i\right]$$

Exactly first *i* transplants

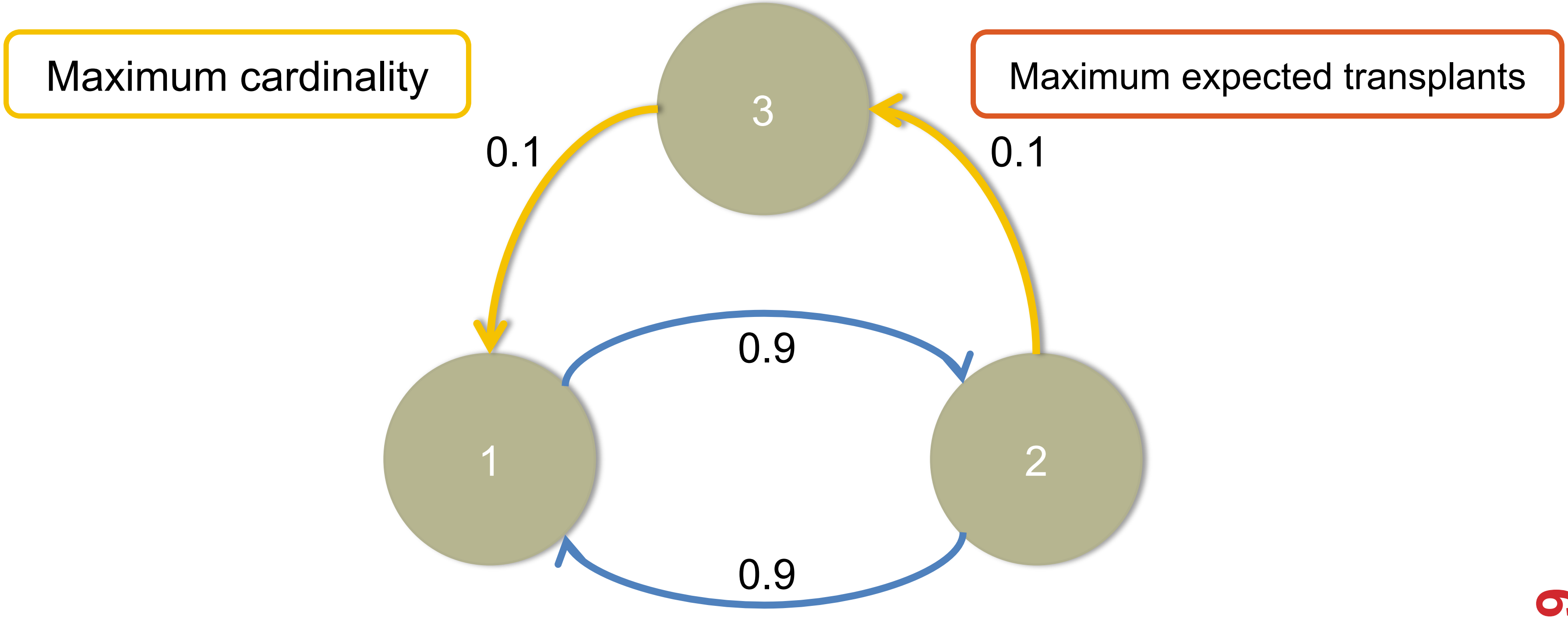
Chain executes in entirety

Cannot simply “reweight by failure probability”

DISCOUNTED CLEARING PROBLEM

(“Best” = max expected cardinality | limited recourse)

Find matching *M with highest discounted utility**



Maximum cardinality

Maximum expected transplants

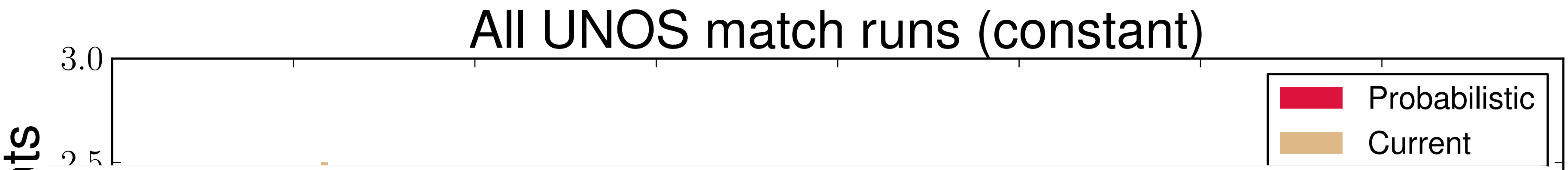
SOLVING THIS NEW PROBLEM

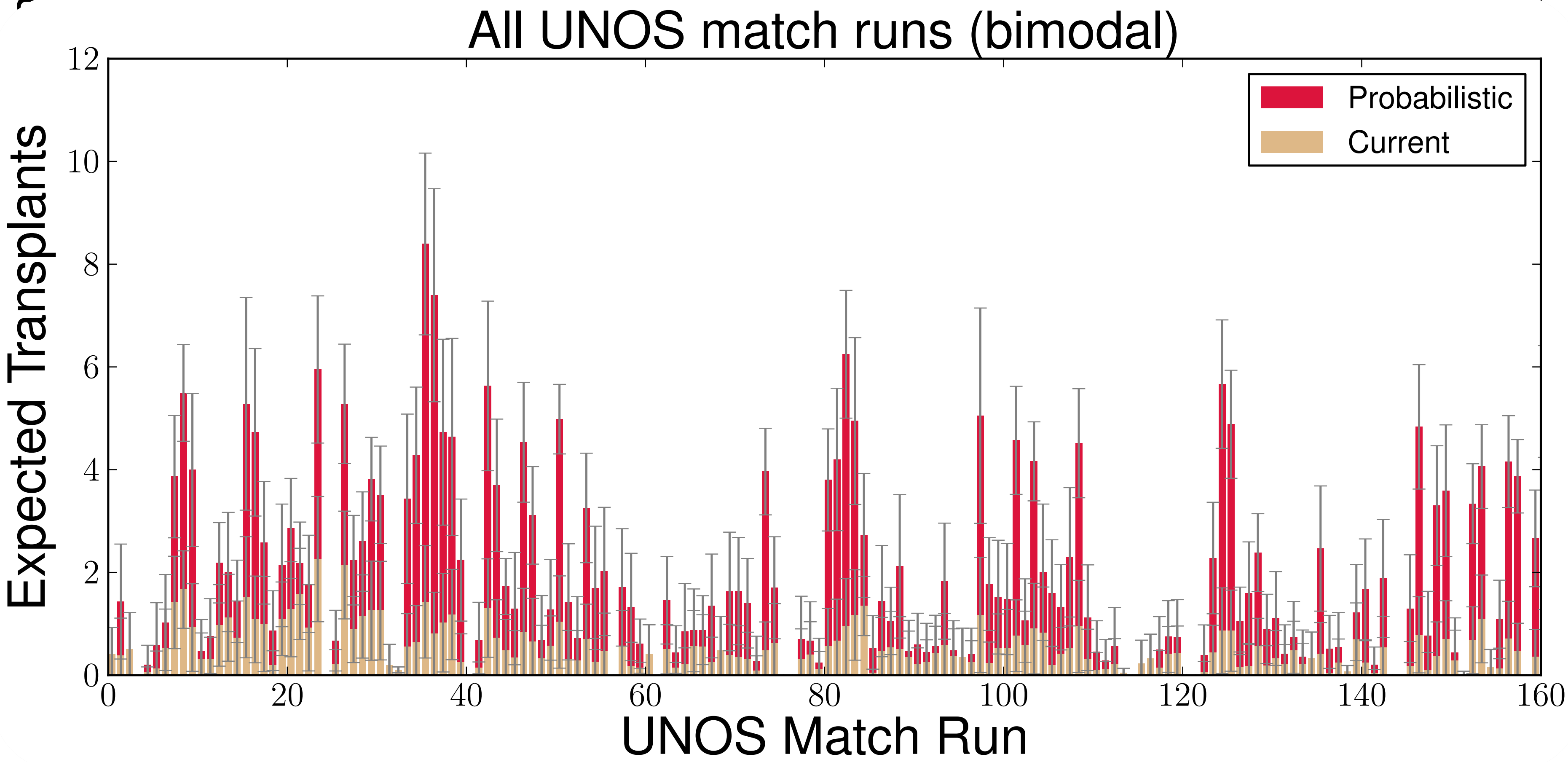
Theorem:
In a sparse random graph model, for any failure probability *p*, w.h.p. there exists a matching that is "linearly better" than *any* max-cardinality matching

Practice: Solved via branch-and-price

- One binary decision variable per cycle/chain
- Upper-bounding is now NP-hard ✗
- Pricing problem is (empirically) much easier ✓

Maybe this is a good idea …





Under discussion for implementation at UNOS

PRE-MATCH EDGE TESTING

Idea: perform a *small amount* of costly testing before a match run to test for (non)existence of edges

- E.g., more extensive medical testing, donor interviews, surgeon interviews, …

Cast as a *stochastic matching* problem:

Given a graph $G(V,E)$, choose subset of edges S such that:

$$|M(S)| \geq (1-\varepsilon)\ |M(E)|$$

Need: “sparse” S, where every vertex has O(1) incident tested edges

GENERAL THEORETICAL RESULTS

Adaptive: select one edge per vertex per *round*, test, repeat

THEOREM

Stochastic matching:
(1-ε) approximation with $O_\varepsilon(1)$ queries per vertex, in $O_\varepsilon(1)$ rounds

Stochastic k-set packing:
(2/k – ε) approximation with $O_\varepsilon(1)$ queries per vertex, in $O_\varepsilon(1)$ rounds

Non-adaptive: select O(1) edges per vertex, test all at once

THEOREM

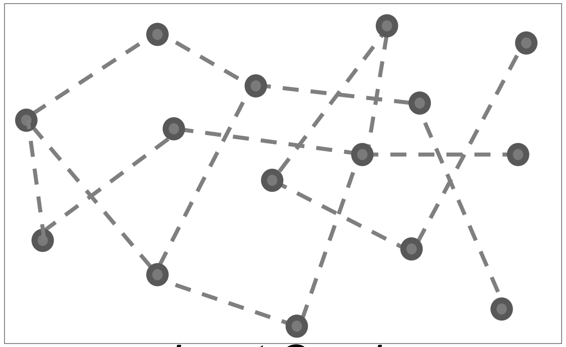
Stochastic matching:
(0.5-ε) approximation with $O_\varepsilon(1)$ queries per vertex, in 1 round

Stochastic k-set packing:
$(2/k – \varepsilon)^2$ approximation with $O_\varepsilon(1)$ queries per vertex, in 1 round

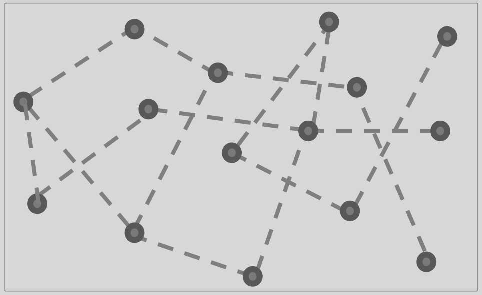
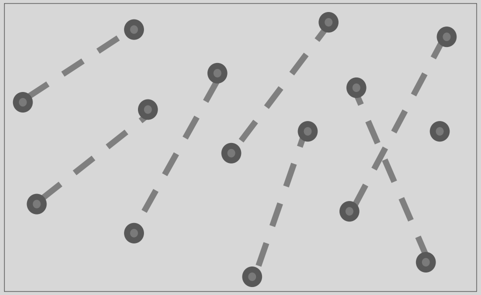
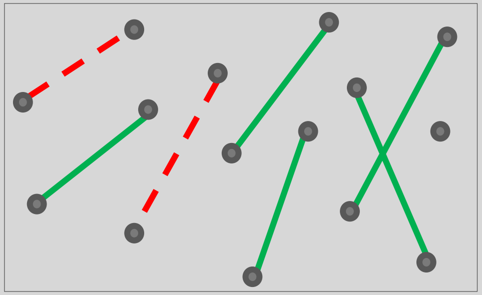
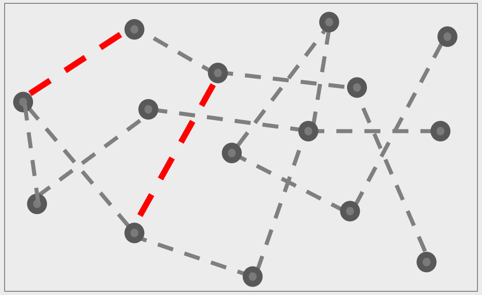
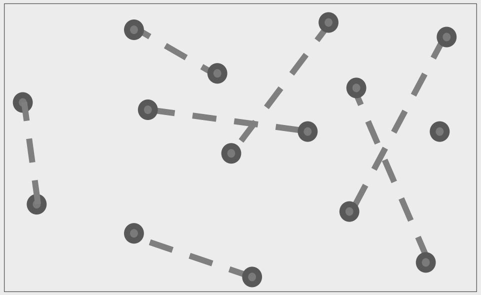
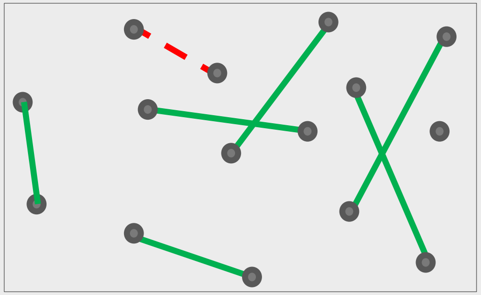
ADAPTIVE ALGORITHM

For *R* rounds, do:

1. Pick a max-cardinality matching *M* in graph *G*, minus already-queried edges that do not exist
2. Query all edges in *M*



Input Graph

r	Base graph	Matching picked	Result of queries
1:			
2:			

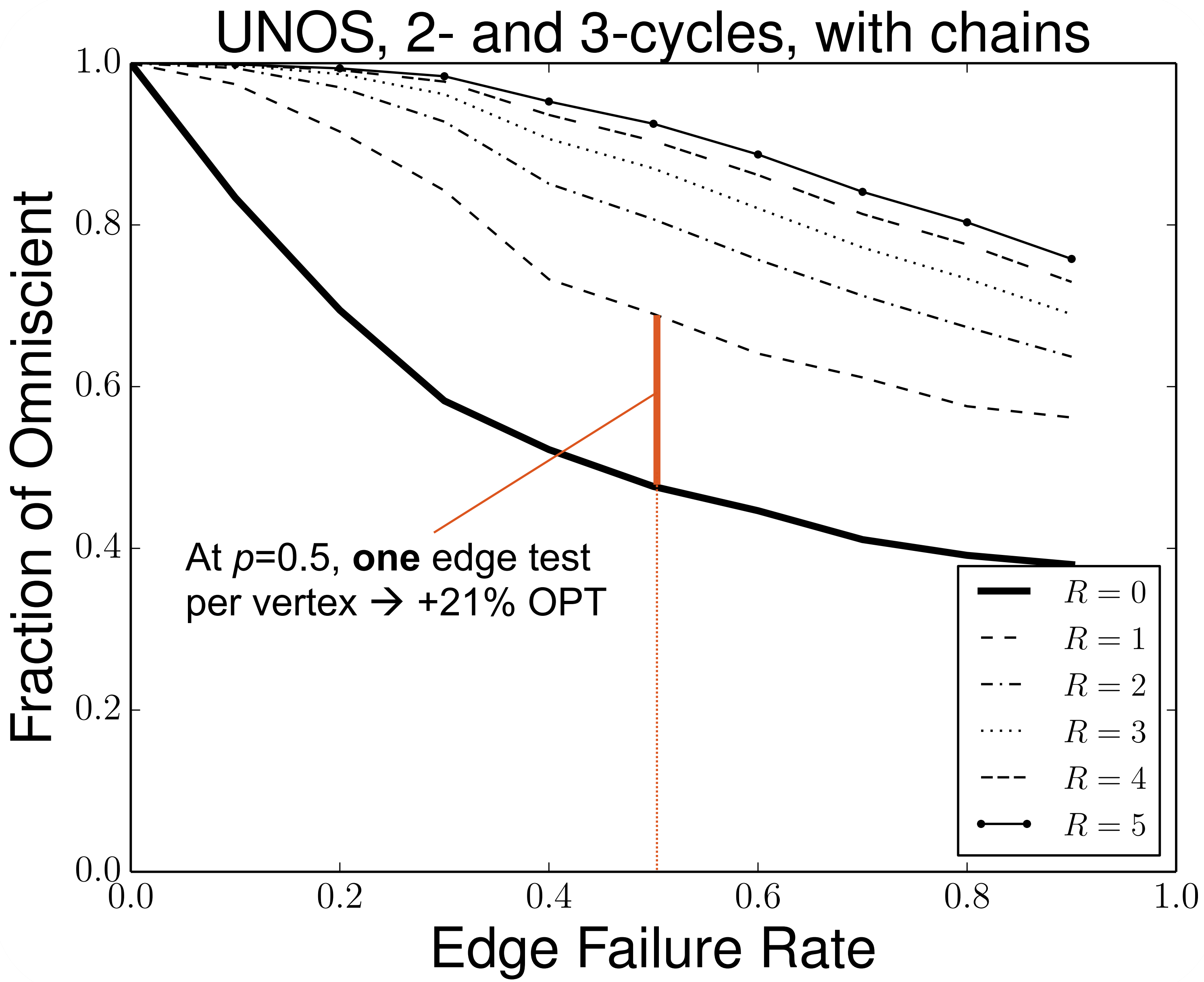
INTUITION FOR ADAPTIVE ALGORITHM

If at any round *r*, the best solution on edges queried so far is small relative to omniscient …

- ... then current structrure admits *large* number of unqueried, disjoint augmenting structures
- For *k*=2, aka normal matching, simply augmenting paths

Augmenting structures might not exist, but can query in parallel in a single round

- Structures are constant size → exist with constant probability
- Structures are disjoint → queries are independent
- → Close a constant gap per round



At *p*=0.5, **one** edge test per vertex → +21% OPT

Even 1 or 2 extra tests would result in a huge lift

In theory and practice, we’re helping the global bottom line by considering post-match failure …

… But can this hurt some individuals?

BALANCING EQUITY AND EFFICIENCY

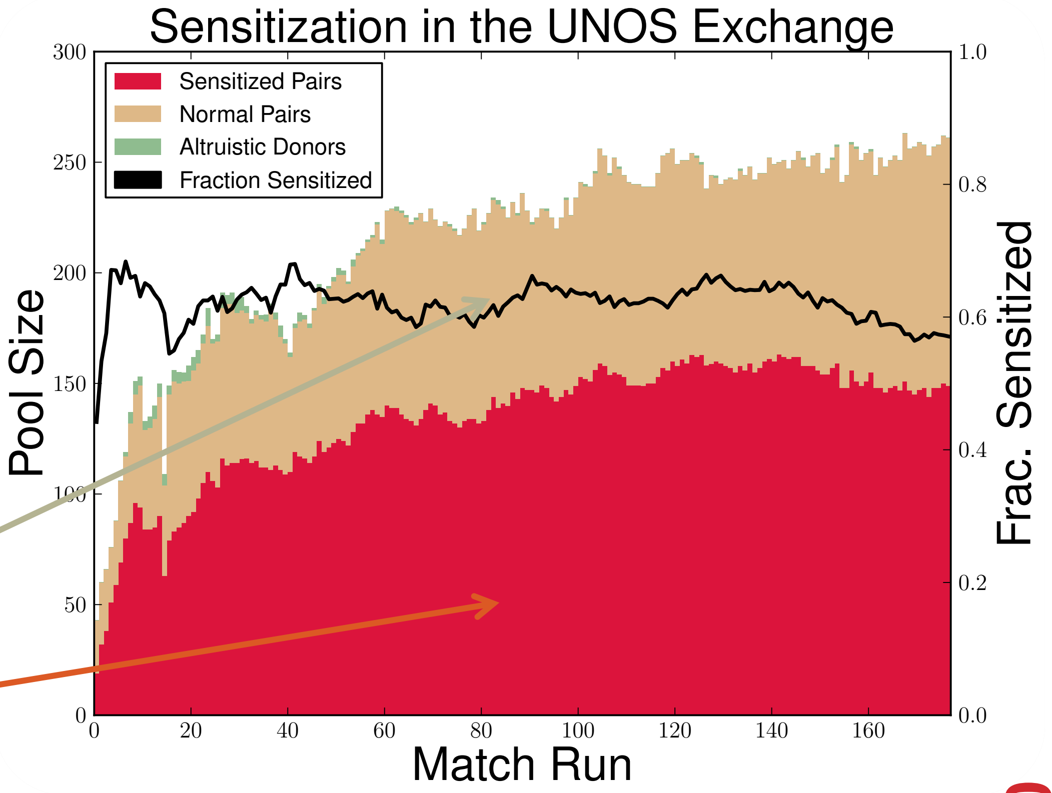
[AAMAS-14, AAAI-15, AAAI-18, Invited to AIJ, u.r. 2018]

With D. McElfresh, A. Procaccia and T. Sandholm

SENSITIZATION AT UNOS

Highly-sensitized patients: unlikely to be compatible with a random donor

- Deceased donor waitlist: 17%
- Kidney exchanges: **much** higher (60%+)



"Easy to match" patients

"Hard to match" patients

PRICE OF FAIRNESS

Efficiency vs. fairness:

- **Utilitarian** objectives may favor certain classes at the expense of marginalizing others
- **Fair** objectives may sacrifice efficiency in the name of egalitarianism

Price of fairness: relative system efficiency loss under a fair allocation [Bertismas, Farias, Trichakis 2011] [Caragiannis et al. 2009]

PRICE OF FAIRNESS IN KIDNEY EXCHANGE

Want a matching M^* that maximizes utility function $u\colon \mathcal{M} \rightarrow \mathbb{R}$

$$M^* = \operatorname*{argmax}_{M \in \mathcal{M}} u(M)$$

Price of fairness: relative loss of match **efficiency** due to **fair** utility function u_f:

$$POF(\mathcal{M}, u_f) = \frac{u(M^*) - u(M_f^*)}{u(M^*)}$$

FROM THEORY TO PRACTICE

We show that the price of fairness is low in theory

$$POF(\mathcal{M}, u_{H \succ L}) \leq {}^{2}/_{33}$$

Fairness criterion: *extremely* strict.

Theoretical assumptions (standard):

- Big, dense graphs ("*n* → ∞")
- Cycles (no chains)
- No post-match failures
- Simplified patient-donor features

What about the price of fairness *in practice*?

TOWARD USABLE FAIRNESS RULES

In healthcare, important to work within (or near to) the constraints of the fielded system

- [Bertsimas, Farias, Trichakis 2013]
- Our experience with UNOS

We now present two (simple, intuitive) rules:

- **Lexicographic**: strict ordering over vertex types
- **Weighted**: implementation of “priority points”

LEXICOGRAPHIC FAIRNESS

Find the best match that includes at least *α* fraction of highly-sensitized patients

***Matching-wide* constraint:**

- Present-day branch-and-price IP solvers rely on an "easy" way to solve the pricing problem
- Lexicographic constraints → pricing problem requires an IP solve, too!

Strong guarantee on match composition …

- … but harder to predict effect on economic efficiency

WEIGHTED FAIRNESS

Value matching a highly-sensitized patient at (1+β) that of a lowly-sensitized patient, β>0

Re-weighting is a preprocess →

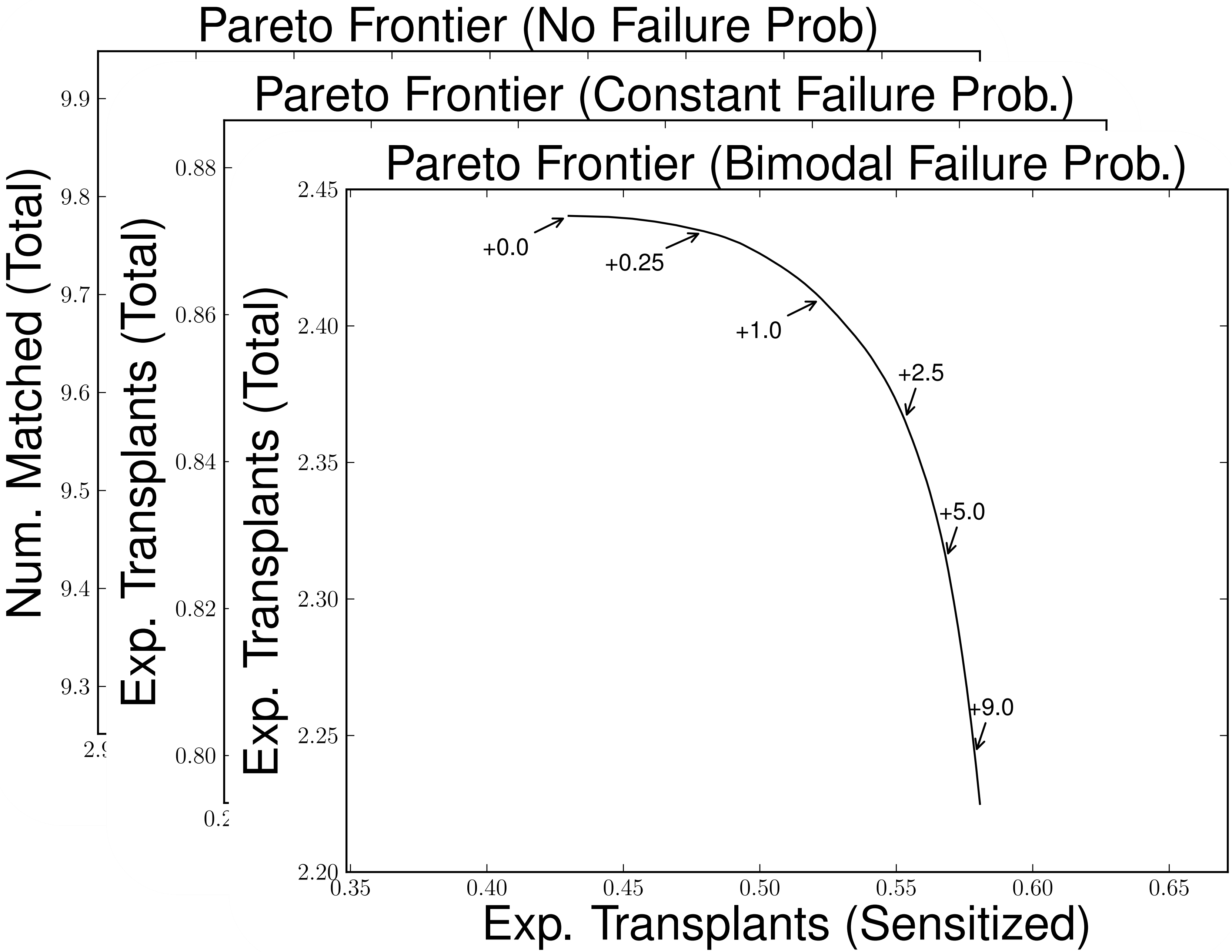
works with all present-day exchange solvers

Difficult to find a "good" β?

- Empirical exploration helps strike a balance

UNOS MATCH RUNS

WEIGHTED FAIRNESS, VARYING FAILURE RATES



Pareto Frontier (No Failure Prob)

Pareto Frontier (Constant Failure Prob.)

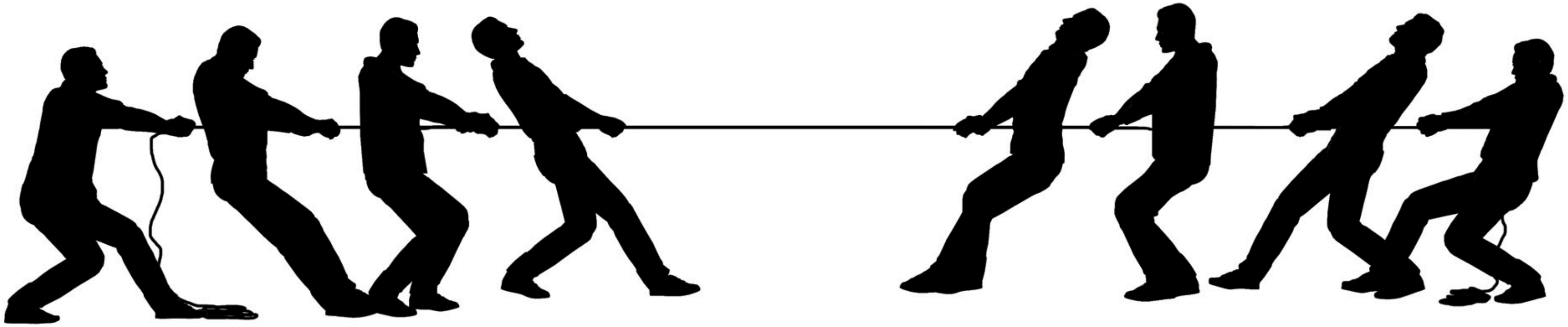
Pareto Frontier (Bimodal Failure Prob.)

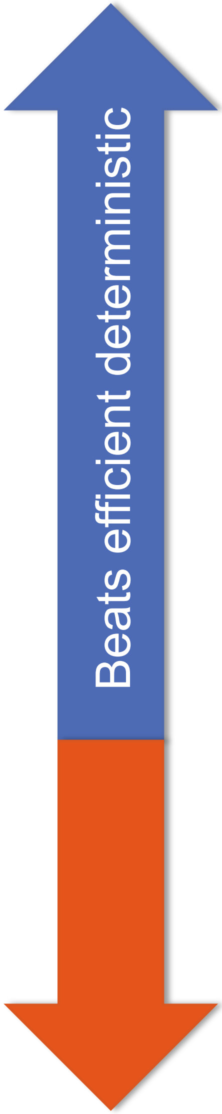
CONTRADICTORY GOALS

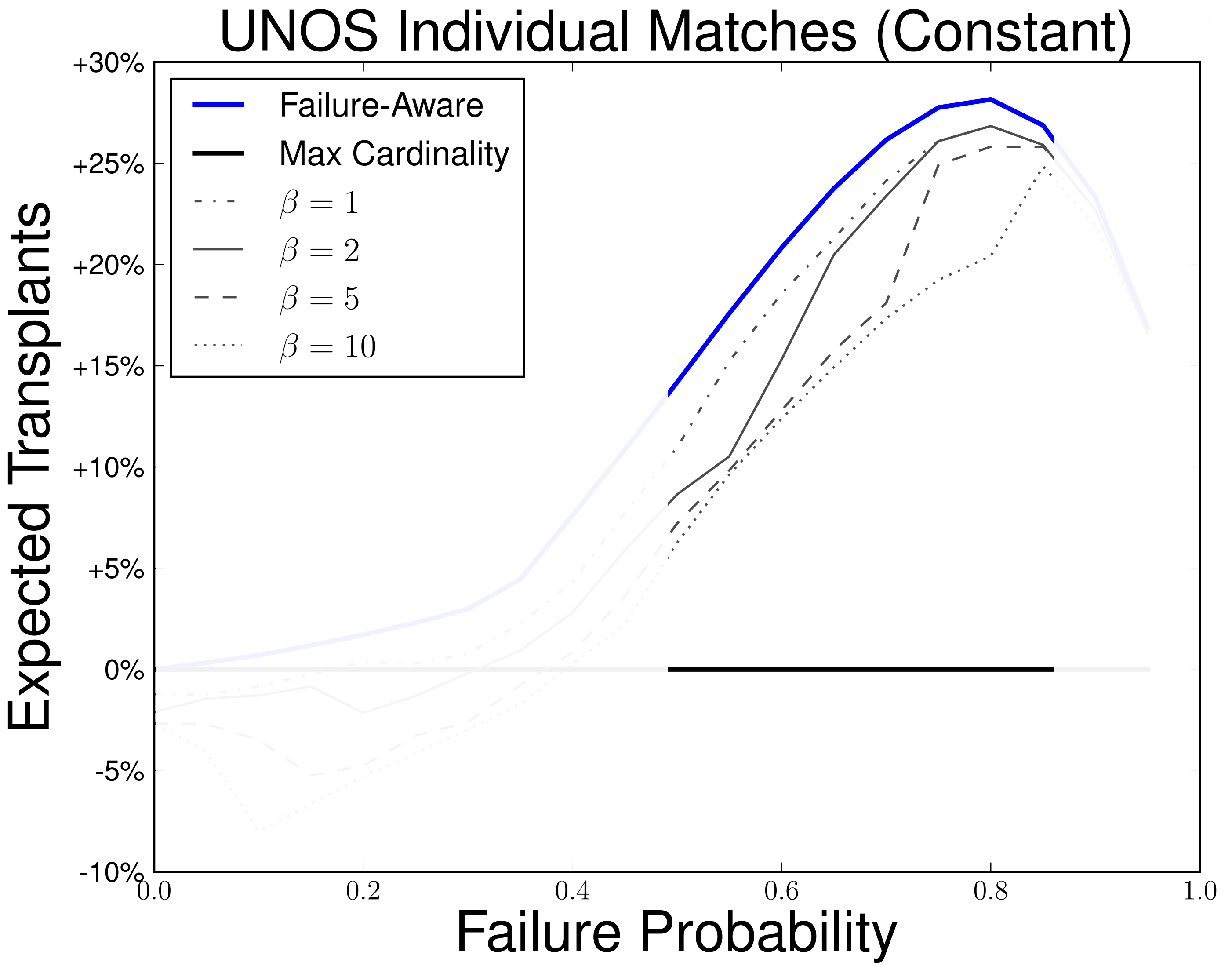
Earlier, we saw failure-aware matching results in tremendous gains in #expected transplants

Gain comes at a price – may further marginalize hard-to-match patients because:

- Highly-sensitized patients tend to be matched in chains
- Highly-sensitized patients may have higher failure rates (in APD data, not in UNOS data)







UNOS runs, weighted fairness, constant probability of failure (x-axis), increase in expected transplants over deterministic matching (y-axis)

Fairness vs. efficiency can be balanced in theory and in practice in a static model …

… But how should we match over time?

LEARNING TO MATCH IN A DYNAMIC ENVIRONMENT

[AAAI-12, AAAI-15, NIPS-15 MLHC, w.p. 2018]

With A. Procaccia and T. Sandholm

DYNAMIC KIDNEY EXCHANGE

Kidney exchange is a naturally dynamic event

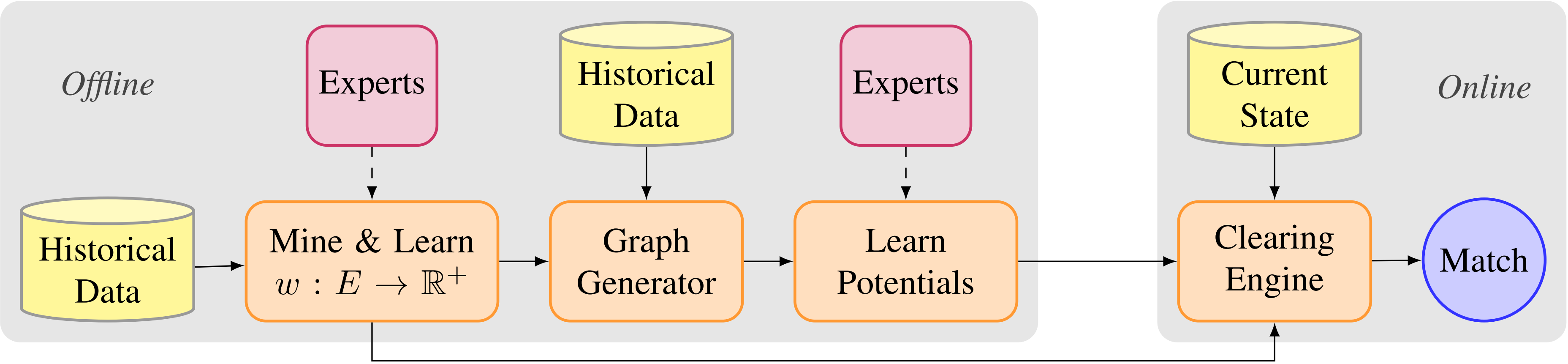
Can be described by the evolution of its graph

- Additions, removals of edges and vertices

Vertex Removal	Edge Removal	Vertex/Edge Add
Transplant, this exchange	Matched, positive crossmatch	Normal entrance
Transplant, deceased donor waitlist	Matched, candidate refuses donor	
Transplant, other exchange ("sniped")	Matched, donor refuses candidate	
Death or illness	Pregnancy, sickness changes HLA	
Altruist runs out of patience		
Bridge donor reneges		

How should we balance matching now versus waiting to match?

FUTUREMATCH: LEARNING TO MATCH IN DYNAMIC ENVIRONMENTS

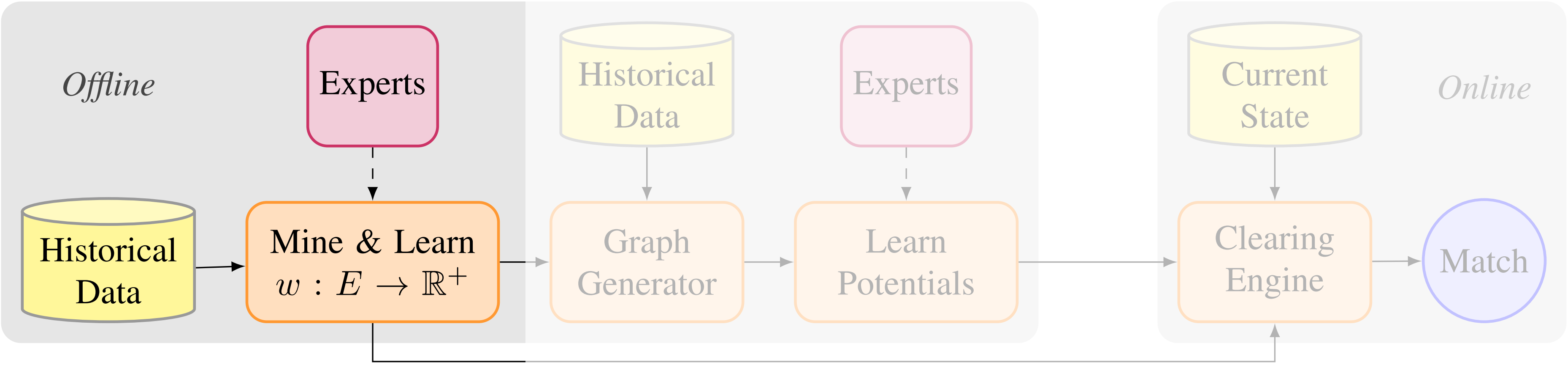


Offline (run once or periodically)

1. Domain expert describes overall goal
2. Take historical data and policy input to learn a weight function *w* for match quality
3. Take historical data and create a graph generator with edge weights set by *w*
4. Using this generator and a realistic exchange simulator, learn potentials for graph elements as a function of the exchange dynamics

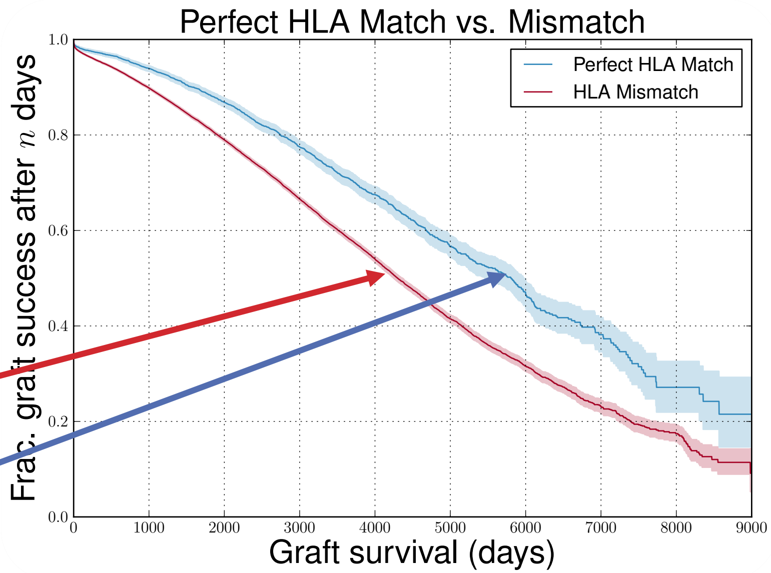
Online (run every match)

1. Combine *w* and potentials to form new edge weights on real input graphs
2. Solve maximum weighted matching and return match

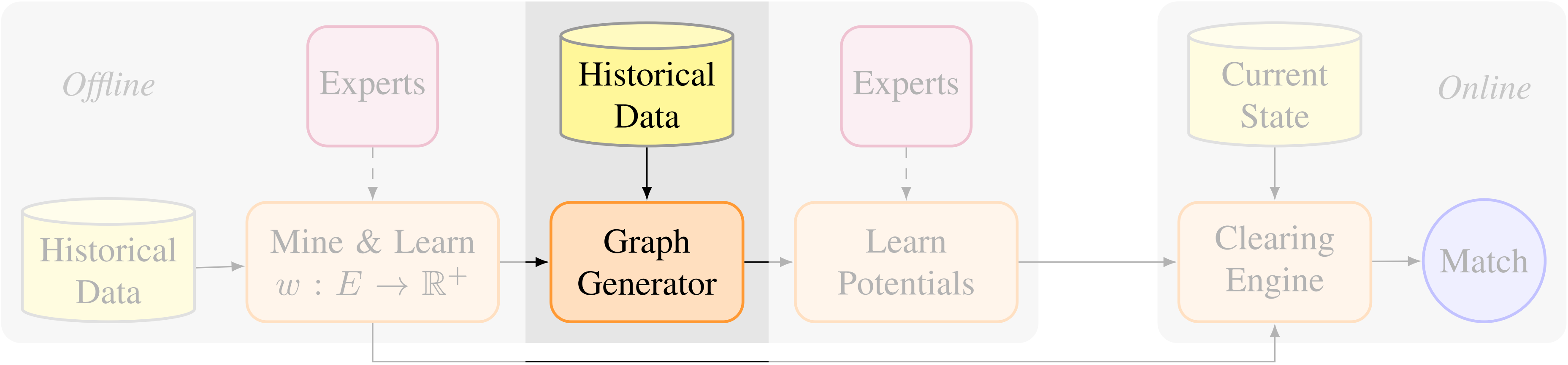


Example objective (MaxLife)

- Maximize aggregate length of time donor organs last in patients …
  - … possibly subject to prioritization schemes, fairness, etc …
- Learn survival rates from all living donations since 1987
  - ~75,000 transplants
- Translate to edge weight

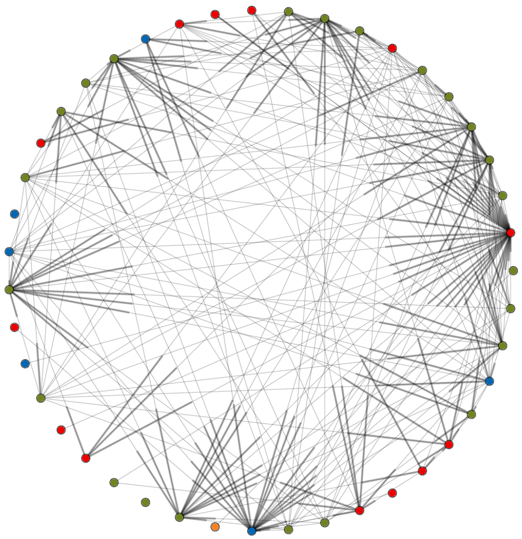


Imperfect HLA match
has worse survival rate than
perfect HLA match

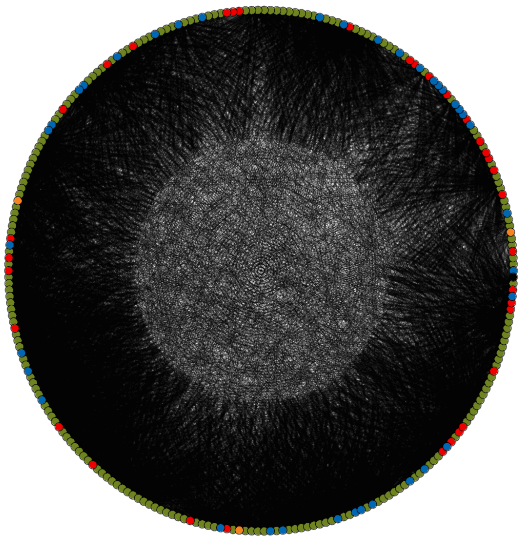


300+ match runs with real UNOS data

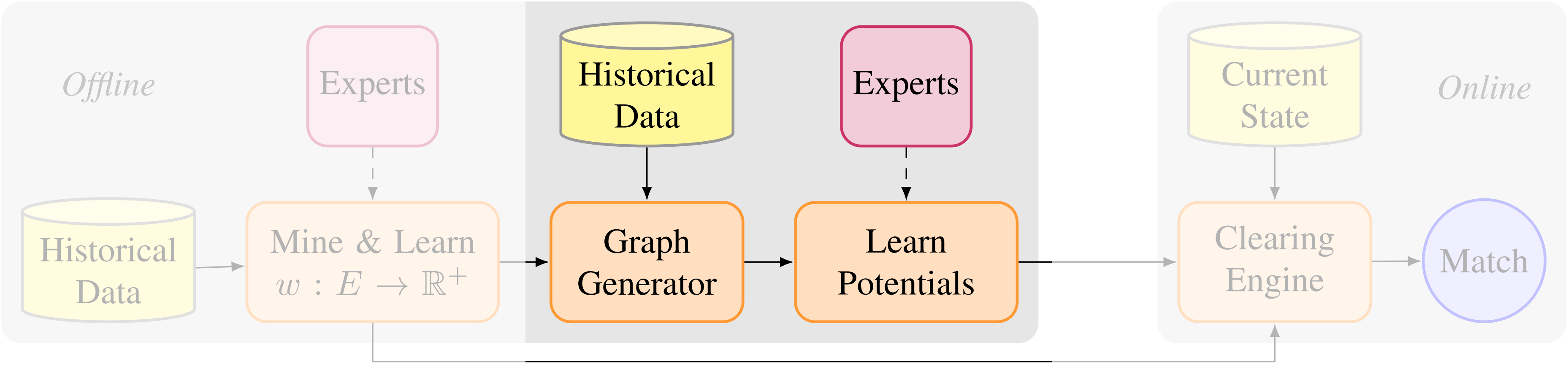
Important to use realistic distribution

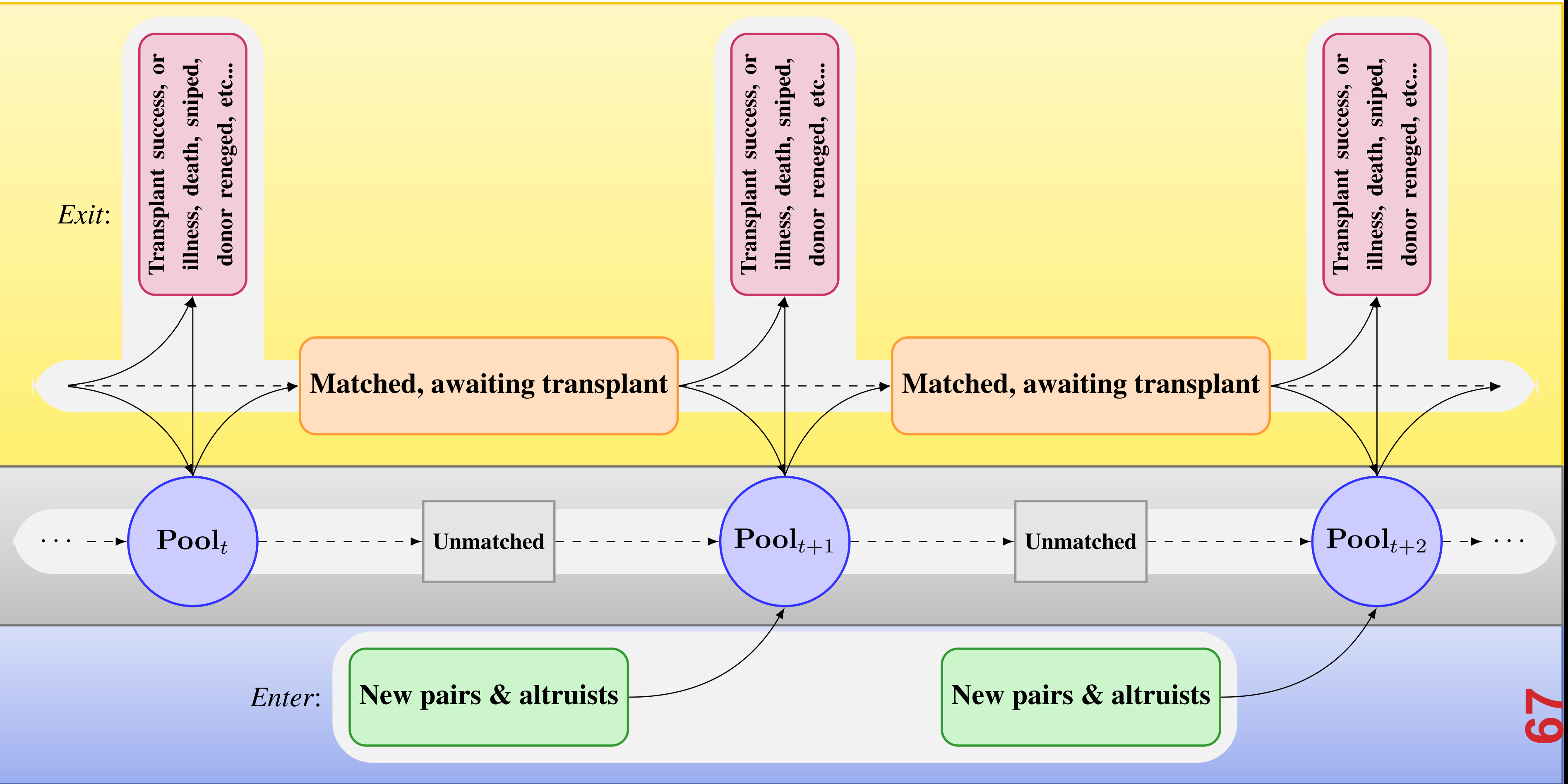


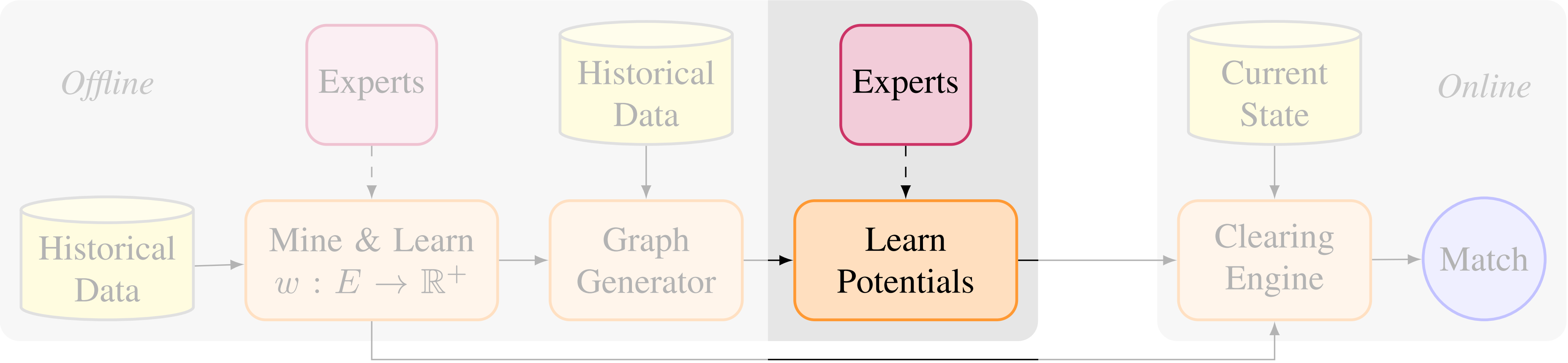
UNOS
(first match run)



UNOS
(recent snapshot)


Offline
Experts
Historical Data
Experts
Current State
Online
Historical Data
Mine & Learn
$w : E \to \mathbb{R}^+$
Graph Generator
Learn Potentials
Clearing Engine
Match


Exit:
Transplant success, or illness, death, sniped, donor reneged, etc...
Transplant success, or illness, death, sniped, donor reneged, etc...
Transplant success, or illness, death, sniped, donor reneged, etc...
Matched, awaiting transplant
Matched, awaiting transplant
· · ·
Pool$_t$
Unmatched
Pool$_{t+1}$
Unmatched
Pool$_{t+2}$
· · ·
Enter:
New pairs & altruists
New pairs & altruists

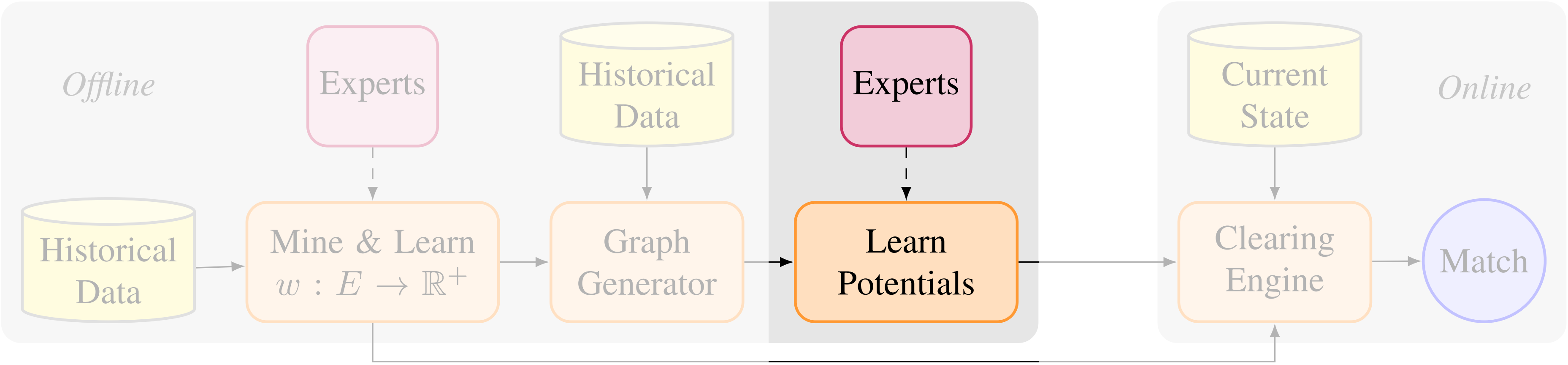


Full optimization problem is very difficult

- Realistic theory is too complex
- Trajectory-based methods do not scale

Approximation idea:

- Associate with each "element type" its **potential** to help objective in the future
- (Must learn these potentials)
- Combine potentials with edge weights, perform myopic maximum utility matching



What is a potential?

Given a set of features *Θ* representing structural elements (e.g., vertex, edge, subgraph type) of a problem:

- The potential P_θ for a type *θ* quantifies the **future usefulness** of that element

E.g., let *Θ* = {O-O, O-A, …, AB-AB, •-O, …, •-AB}

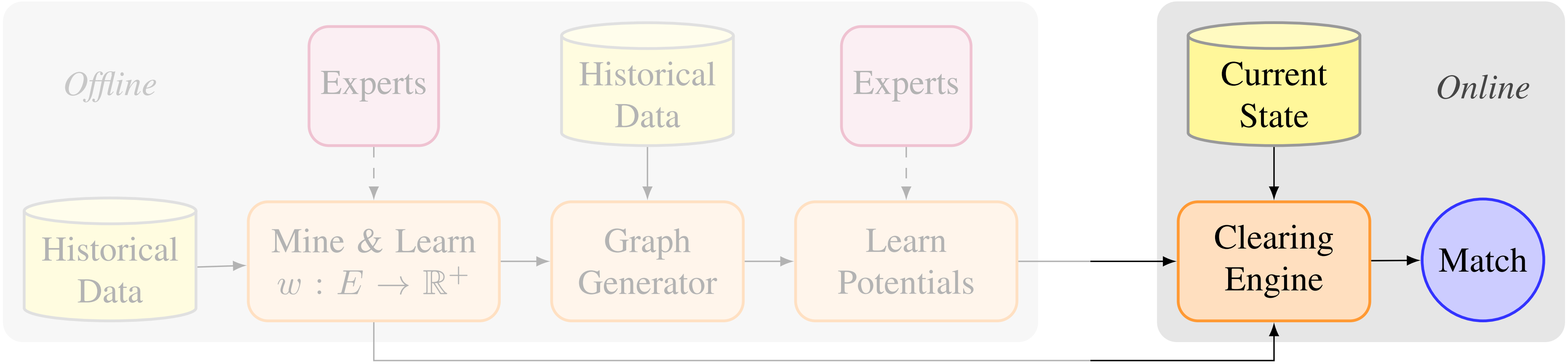
- 16 patient-donor types, 4 altruist types
- O-donors better than A-donors, so: $P_{\bullet\text{-}O} > P_{\bullet\text{-}A}$

Heavy one-time computation to learn potential of each type *θ* – we use SMAC

[Hutter Hoos Leyton-Brown 2011]

PITTSBURGH SUPERCOMPUTING CENTER

IBM



Online:

Adjust solver to take potentials into account at runtime

- E.g., $P_{\bullet\text{-}O}$ = 2.1 and $P_{O\text{-}AB}$ = 0.1
- Edges between O-altruist and O-AB pair has weight:
  1 – 0.5(2.1+0.1) = -0.1
- Chain must be long enough to offset negative weight

Also take into account learned weight function *w*

Edge weight preprocess → no or low runtime hit!

PITTSBURGH SUPERCOMPUTING CENTER

EXPERIMENTAL RESULTS & IMPACT

Best Data-Intensive Application/Use
HPC wire
Readers' Choice Awards 2014

Presented at Supercomputing
Tied with IBM Watson

We show it is possible to:

- Increase overall #transplants a lot at a (much) smaller decrease in #marginalized transplants
- Increase #marginalized transplants a lot at no or very low decrease in overall #transplants
- Increase both #transplants and #marginalized

Sweet spot depends on distribution:

- Luckily, we can generate – and learn from – realistic families of graphs!

FutureMatch now used for policy recommendations at UNOS

THE CUTTING EDGE

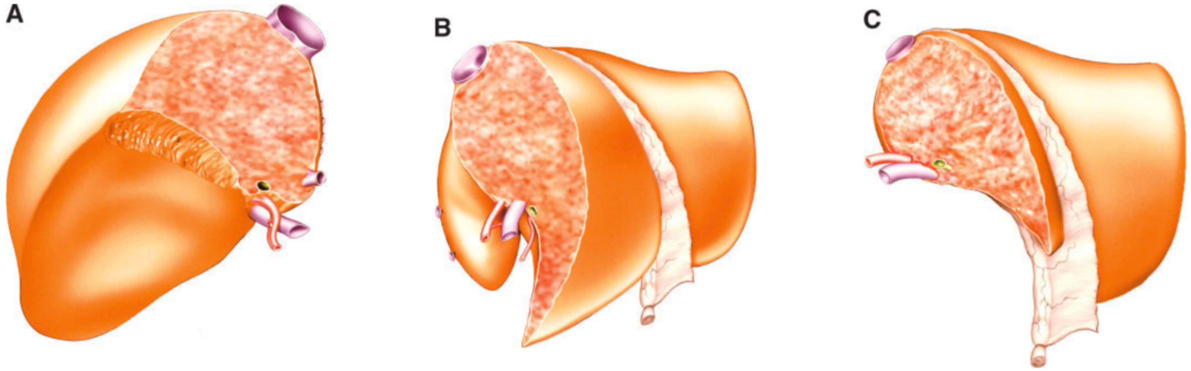
MOVING BEYOND KIDNEYS: LIVERS [Ergin, Sönmez, Ünver *w.p. 2015*]

Similar matching problem (mathematically)

Right Lobe
Segments 5-8

Left + Caudate Lobes
Segments 2-4

Left Lateral Segment
Segments 2-3



Donor Mortality: 0.5%
Size: 60%
Most risky!

Donor Mortality: 0.1%
Size: 40%
Often too small

Donor Mortality: Rare
Size: 20%
Only pediatric

[Sönmez 2014]

Right lobe is biggest but riskiest; exchange may reduce right lobe usage and increase transplants

MOVING BEYOND KIDNEYS: MULTI-ORGAN EXCHANGE

[Dickerson Sandholm AAAI-14, JAIR-16]

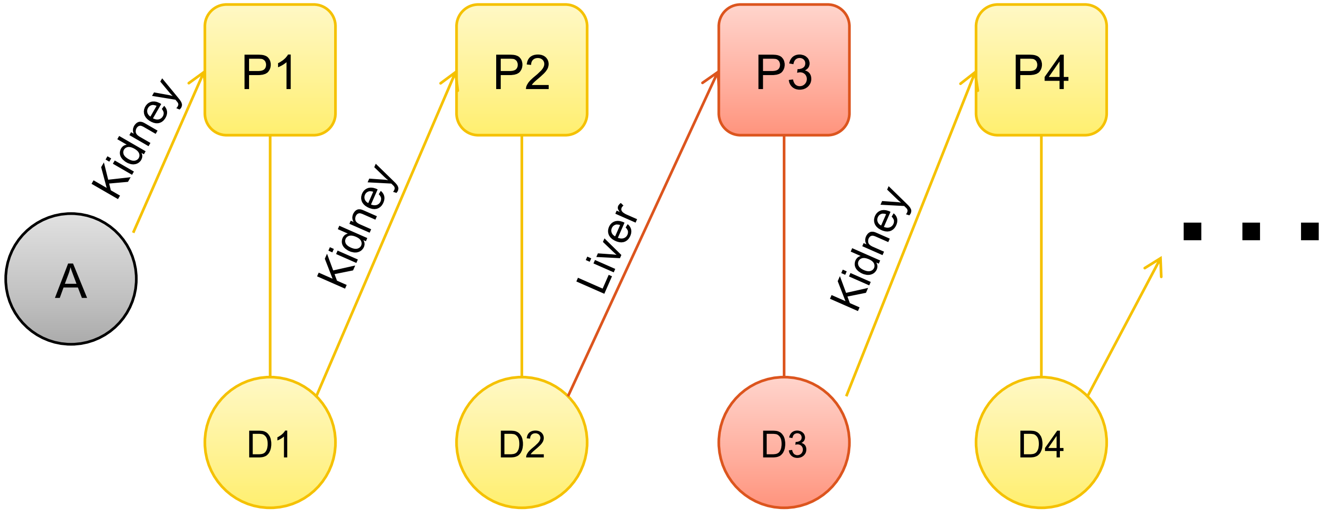
Chains are great! [Anderson et al. 2015, Ashlagi et al. 2014, Rees et al. 2009]

Kidney transplants are “easy” and popular:

- Many altruistic donors

Liver transplants: higher mortality, morbidity:

- (Essentially) no altruistic donors



SPARSE GRAPH, MANY ALTRUISTS

n_K kidney pairs in graph D_K; $n_L = \gamma n_K$ liver pairs in graph D_L

Number of altruists t(n_K)

Constant $p_{K\rightarrow L} > 0$ of kidney donor willing to give liver

Constant cycle cap z

Theorem

Assume $t(n_K) = \beta n_K$ for some constant β>0. Then, with probability 1 as $n_K \rightarrow \infty$,

Any efficient matching on D = join(D_K,D_L) matches $\Omega(n_K)$ more pairs than the aggregate of efficient matchings on D_K and D_L.

Building on [Ashlagi et al. 2012]

INTUITION

Find a linear number of "good cycles" in D_L that are length > z

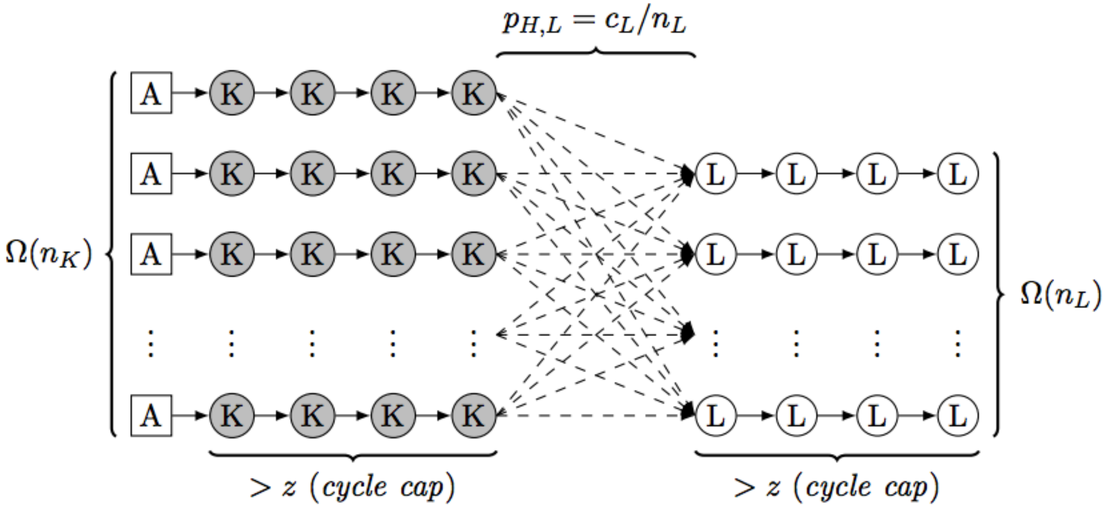
- Good cycles = isolated path in highly-sensitized portion of pool and exactly one node in low portion

Extend chains from D_K into the isolated paths (aka can't be matched otherwise) in D_L, of which there are linearly many

- Have to worry about $p_{K \rightarrow L}$, and compatibility between vertices

Show that a subset of the dotted edges below results in a linear-in-number-of-altruists max matching

- → linear number of D_K chains extended into D_L
- → linear number of previously unmatched D_L vertices matched



SPARSE GRAPH, FEW ALTRUISTS

n_K kidney pairs in graph D_K; $n_L = \gamma n_K$ liver pairs in graph D_L

Number of altruists *t* – no longer depends on n_K!

λ is frac. lowly-sensitized

Constant cycle cap *z*

Theorem

Assume constant *t*. Then there exists λ' > 0 s.t. for all λ < λ'

Any efficient matching on $D = \text{join}(D_K, D_L)$ matches $\Omega(n_K)$ more pairs than the aggregate of efficient matchings on D_K and D_L.

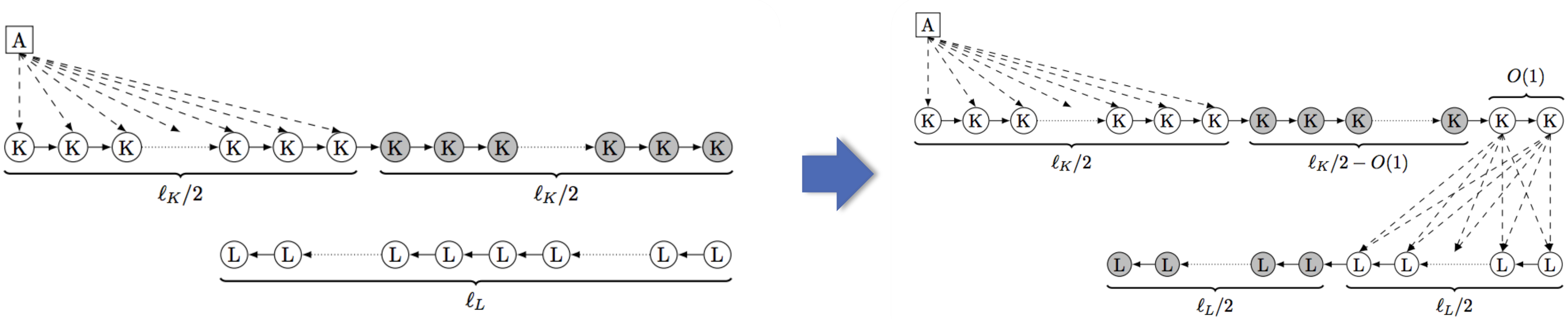
With constant positive probability. *Building on [Ashlagi et al. 2012]*

INTUITION

For large enough λ (i.e., lots of sensitized patients), there exist pairs in D_K that can't be matched in short cycles, thus only in chains

- Same deal with D_L, except there are no chains

Connect a long chain (+altruist) in D_K into an unmatchable long chain in D_L, such that a linear number of D_L pairs are now matched



ETHICAL ISSUES EXIST: BUT, THIS RECENTLY HAPPENED!

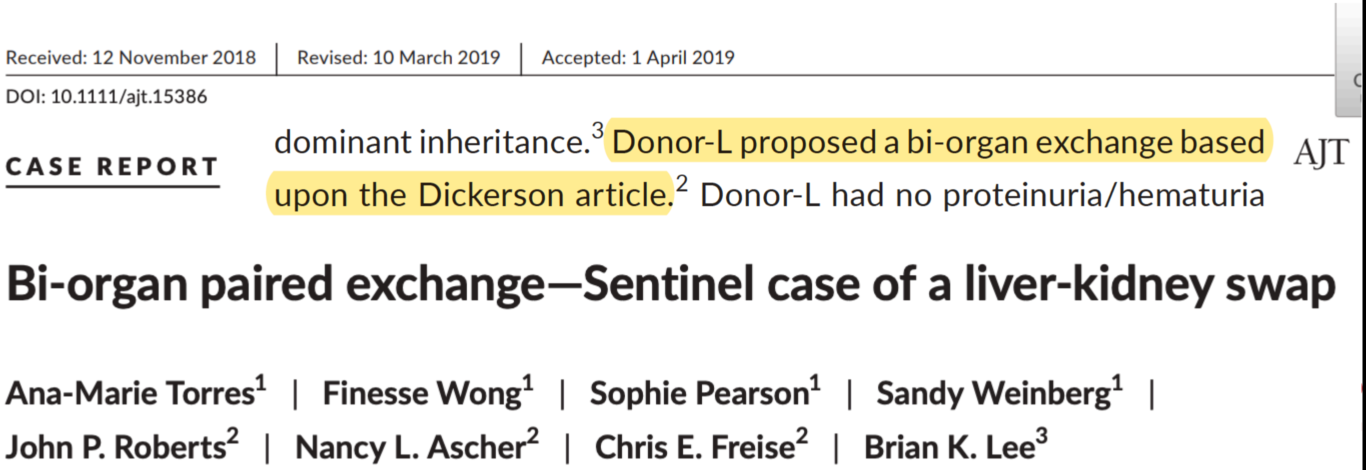
Patient-donor pairs are now exchanging different goods

600% incremental increase in mortality risk for liver vs. kidney donor

1/3000 risk of death for kidney donors [Muzaale et al. Gastroenterology 2012]

1/500 risk of death for liver donors [Cheah et al. Liver Transplantation 2013]

AJT



Received: 12 November 2018 | Revised: 10 March 2019 | Accepted: 1 April 2019

DOI: 10.1111/ajt.15386

CASE REPORT

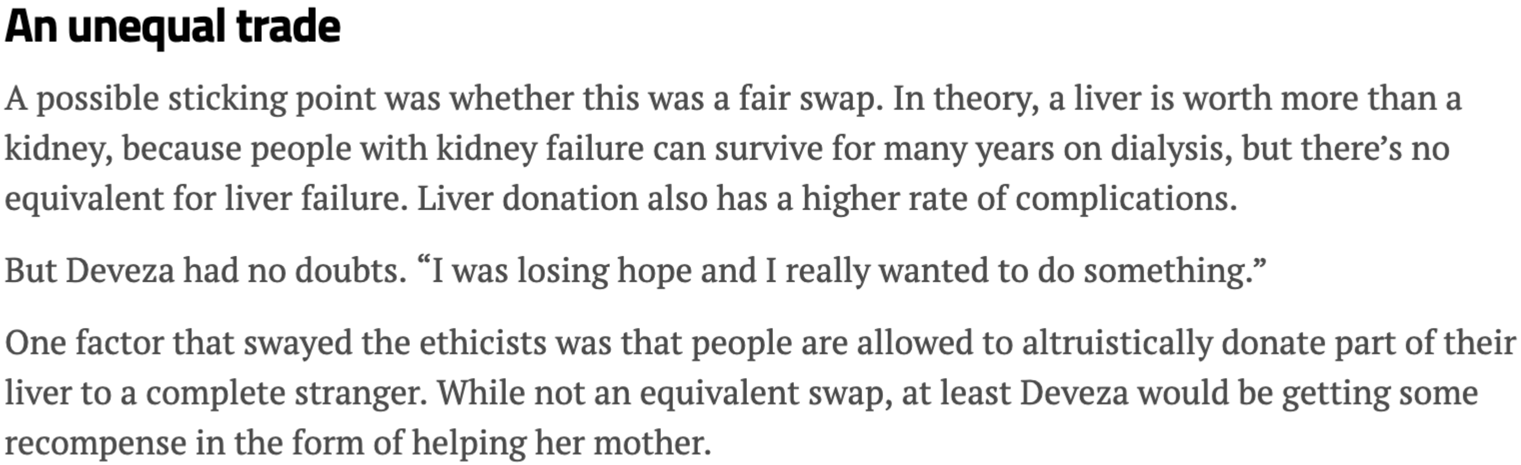
dominant inheritance.[3] Donor-L proposed a bi-organ exchange based upon the Dickerson article.[2] Donor-L had no proteinuria/hematuria

AJT

Bi-organ paired exchange—Sentinel case of a liver-kidney swap

Ana-Marie Torres[1] | Finesse Wong[1] | Sophie Pearson[1] | Sandy Weinberg[1] | John P. Roberts[2] | Nancy L. Ascher[2] | Chris E. Freise[2] | Brian K. Lee[3]

REAL-WORLD REASONING ABOUT ETHICS



An unequal trade

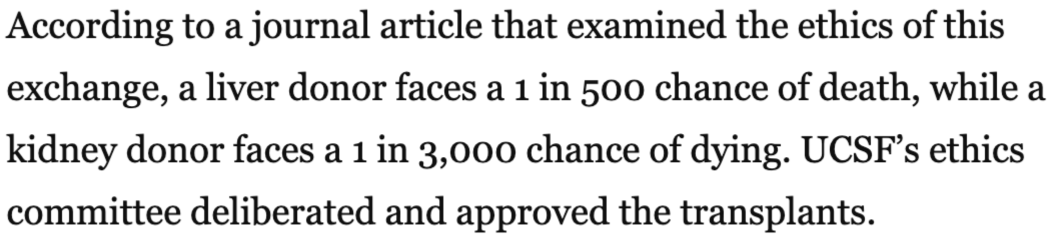
A possible sticking point was whether this was a fair swap. In theory, a liver is worth more than a kidney, because people with kidney failure can survive for many years on dialysis, but there's no equivalent for liver failure. Liver donation also has a higher rate of complications.

But Deveza had no doubts. "I was losing hope and I really wanted to do something."

One factor that swayed the ethicists was that people are allowed to altruistically donate part of their liver to a complete stranger. While not an equivalent swap, at least Deveza would be getting some recompense in the form of helping her mother.

NewScientist

The Washington Post



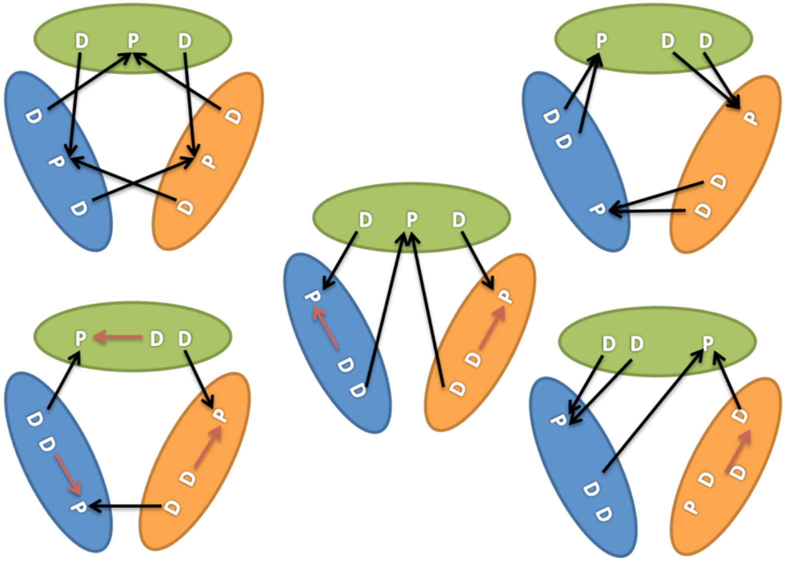
According to a journal article that examined the ethics of this exchange, a liver donor faces a 1 in 500 chance of death, while a kidney donor faces a 1 in 3,000 chance of dying. UCSF's ethics committee deliberated and approved the transplants.

A daughter's gift to her mother saves two lives – The Washington Post – 11 May 2019
My liver, your kidney: The world's first non-identical organ swap – New Scientist – 9 May 2019

MOVING BEYOND KIDNEYS: LUNGS [Ergin, Sönmez, Ünver *w.p. 2014*]

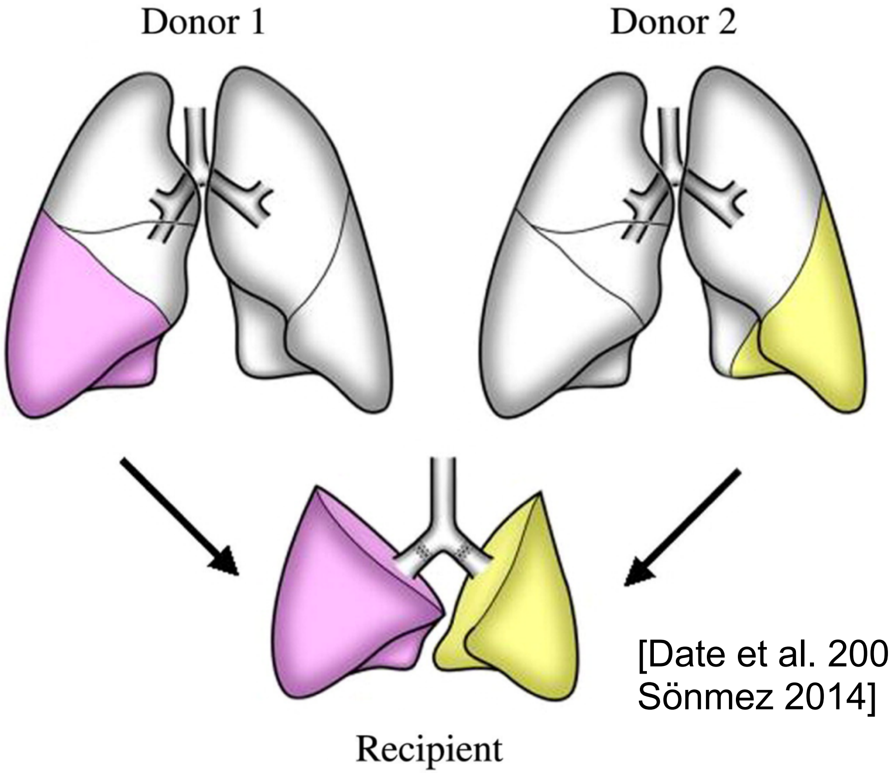
Fundamentally different matching problem

- **Two** donors needed



3-way lung exchange configurations

(Compare to the single configuration for a "3-cycle" in kidney exchange.)



[Date et al. 2005; Sönmez 2014]

OTHER RECENT & ONGOING RESEARCH IN THIS SPACE

Dynamic matching theory with a kidney exchange flavor:

- Akbarpour et al., “Thickness and Information in Dynamic Matching Markets”
- Anderson et al., “A dynamic model of barter exchange”
- Ashlagi et al., “On matching and thickness in heterogeneous dynamic markets”
- Das et al., “Competing dynamic matching markets”

Mechanism design:

- Blum et al. “Opting in to optimal matchings”

Not “in the large” random graph models:

- Ding et al., “A non-asymptotic approach to analyzing kidney exchange graphs

IS LIFE ALWAYS SO (NP-)HARD?

ONE SIMPLE ASSUMPTION COMPLEXITY THEORY HATES!

[Dickerson Kazachkov Procaccia Sandholm arxiv:1605.07728]

- **Observation: real graphs are constructed from a few thousand if statements**
  - If the patient and donor have compatible blood types …
  - ... and if they are compatible on 61 tissue type features ...
  - ... and if their insurances match, and ages match, and ...
  - ... then draw a directed edge; otherwise, don’t

THEOREM

Given a constant number of if statements and a constant cycle cap, the clearing problem is in **polynomial time**

- **Hypothesis: real graphs can be represented by a small constant number of bits per vertex – we’ll test later**

A NEW MODEL FOR KIDNEY EXCHANGE

[Dickerson et al. arxiv:1605.07728]

- **Graph G = (V, E) with patient-donor pair v_i in V with**
  - Attribute vectors d_i and p_i such that the qth element of d_i (resp. p_i) takes on one of a fixed number of types
  - E.g., d_i^q or p_i^q takes a blood type in {O, A, B, AB}
  - Call Θ the set of all possible "types" of d and p
- **Then, given compatibility function $f : \Theta \times \Theta \rightarrow \{0,1\}$ that uniquely determines if an edge between d_i and p_j exists**
  - We can create any compatibility graph (for large enough vectors in D and P)
- **(Altruists are patient-donor pairs where the "patient" is compatible with all donors → chains are now cycles)**

CLEARING IS NOW IN POLYNOMIAL TIME

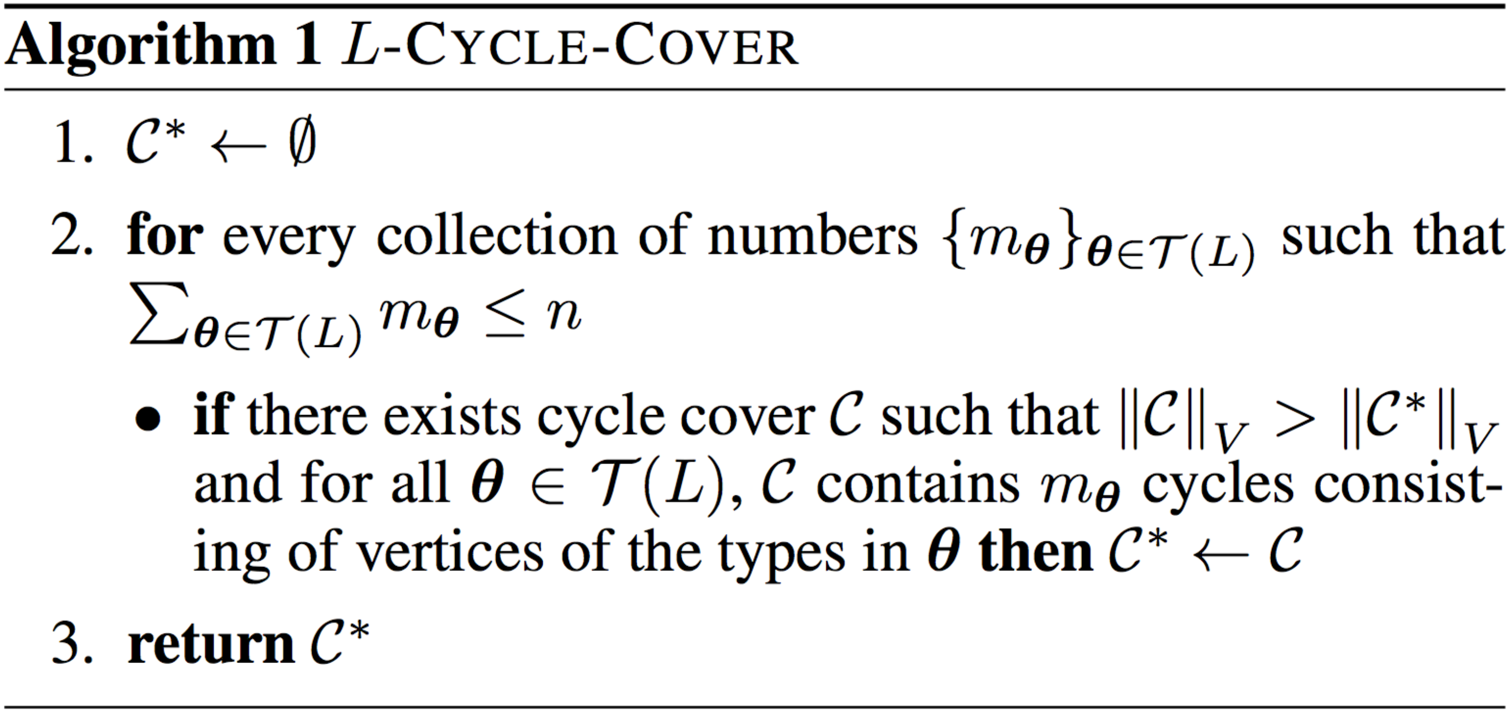
THEOREM

Given constant L and $|\Theta|$,
the clearing problem is in polynomial time

- **Let $f(\theta,\theta') = 1$ if there is a directed edge from a donor with type θ to a patient with type θ'**
- **For all $\theta = (<\theta_{1,p},\theta_{1,d}> \ldots, <\theta_{r,p},\theta_{r,d}>)$ in Θ^{2r} let**
  $f_C(\theta) = 1$ if $f(\theta_{t,d},\theta_{t+1,p}) = 1$ and $f(\theta_{r,d},\theta_{1,p}) = 1$
- **Given cycle cap L, define**
  $T(L) = \{ \theta \text{ in } \Theta^{2r} : r \leq L \text{ and } f_C(\theta) = 1 \}$

CLEARING IS NOW IN POLYNOMIAL TIME

- **T(*L*) is all vectors of types that create feasible cycles of length up to *L***

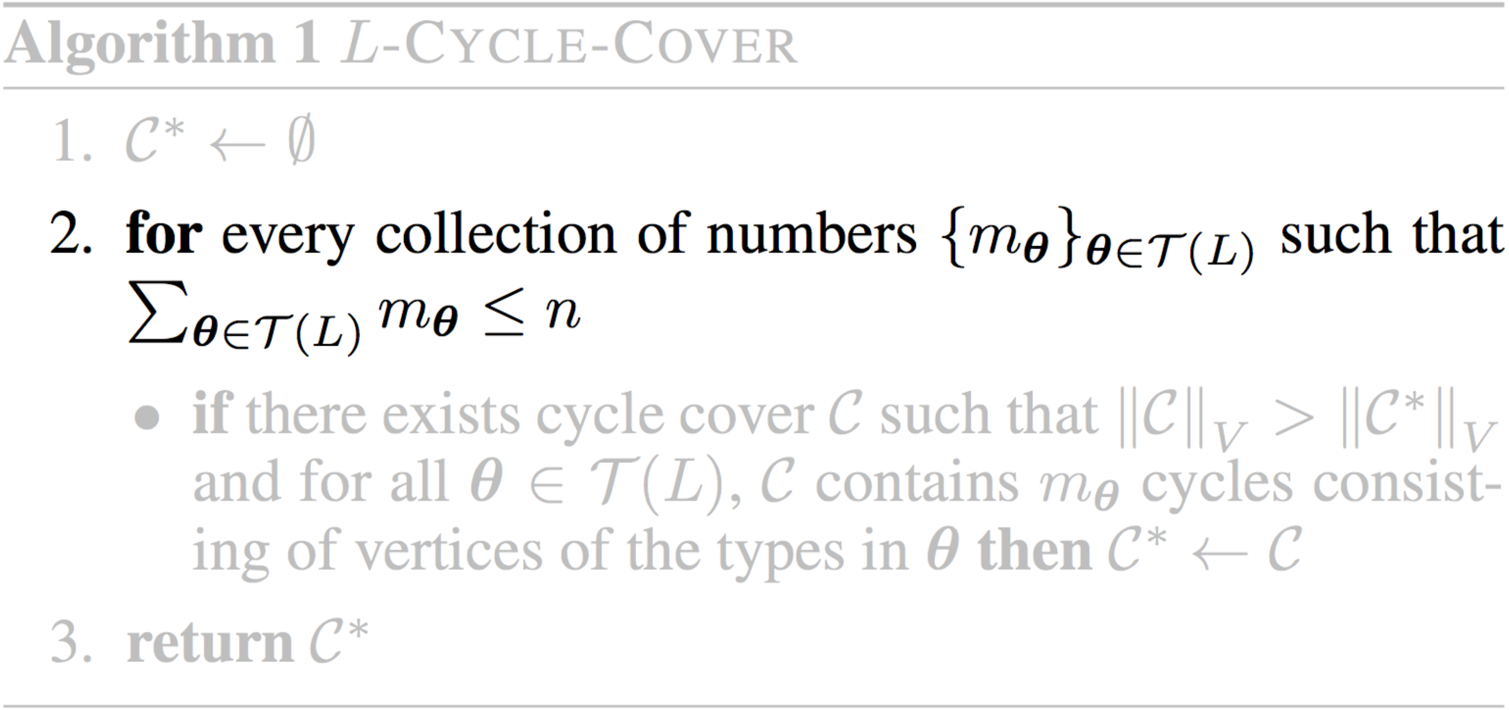


Algorithm 1 L-CYCLE-COVER

```
1. $\mathcal{C}^* \leftarrow \emptyset$
2. for every collection of numbers $\{m_{\boldsymbol{\theta}}\}_{\boldsymbol{\theta} \in \mathcal{T}(L)}$ such that $\sum_{\boldsymbol{\theta} \in \mathcal{T}(L)} m_{\boldsymbol{\theta}} \leq n$
    • if there exists cycle cover $\mathcal{C}$ such that $\|\mathcal{C}\|_V > \|\mathcal{C}^*\|_V$ and for all $\boldsymbol{\theta} \in \mathcal{T}(L)$, $\mathcal{C}$ contains $m_{\boldsymbol{\theta}}$ cycles consisting of vertices of the types in $\boldsymbol{\theta}$ then $\mathcal{C}^* \leftarrow \mathcal{C}$
3. return $\mathcal{C}^*$
```

CLEARING IS NOW IN POLYNOMIAL TIME

- **Each set $\{m_\theta\}$ says we have $m_{\theta 1}$ cycles of type θ_1, $m_{\theta 2}$ cycles of θ_2, …, $m_{\theta |T(L)|}$ cycles of $\theta_{|T(L)|}$, constrained to at most *n* cycles total**

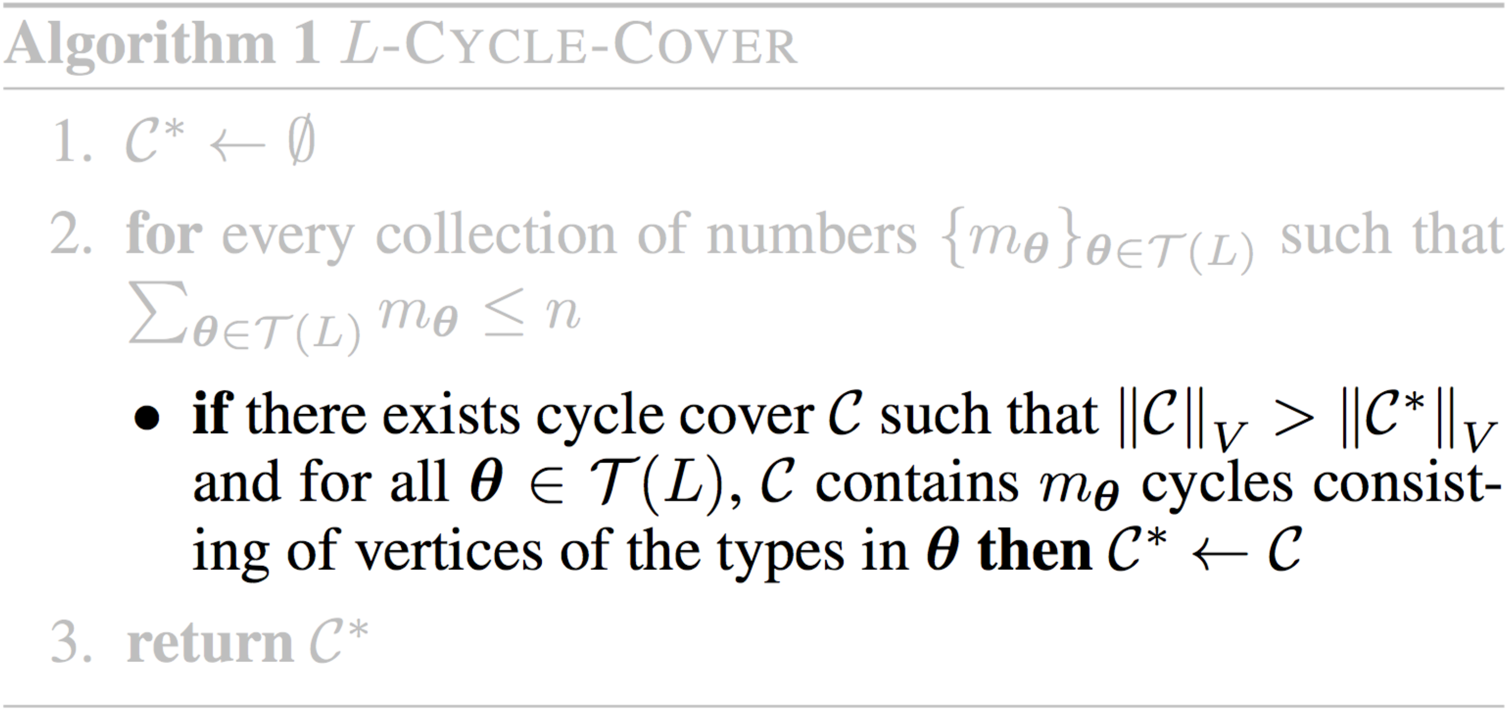


Algorithm 1 L-CYCLE-COVER

```
1. $\mathcal{C}^* \leftarrow \emptyset$
2. for every collection of numbers $\{m_{\boldsymbol{\theta}}\}_{\boldsymbol{\theta} \in \mathcal{T}(L)}$ such that $\sum_{\boldsymbol{\theta} \in \mathcal{T}(L)} m_{\boldsymbol{\theta}} \leq n$
   • if there exists cycle cover $\mathcal{C}$ such that $\|\mathcal{C}\|_V > \|\mathcal{C}^*\|_V$ and for all $\boldsymbol{\theta} \in \mathcal{T}(L)$, $\mathcal{C}$ contains $m_{\boldsymbol{\theta}}$ cycles consisting of vertices of the types in $\boldsymbol{\theta}$ then $\mathcal{C}^* \leftarrow \mathcal{C}$
3. return $\mathcal{C}^*$
```

CLEARING IS NOW IN POLYNOMIAL TIME

- **Check to see if this collection is a legal cycle cover – just check that each type θ isn't used too many times in m_θ**

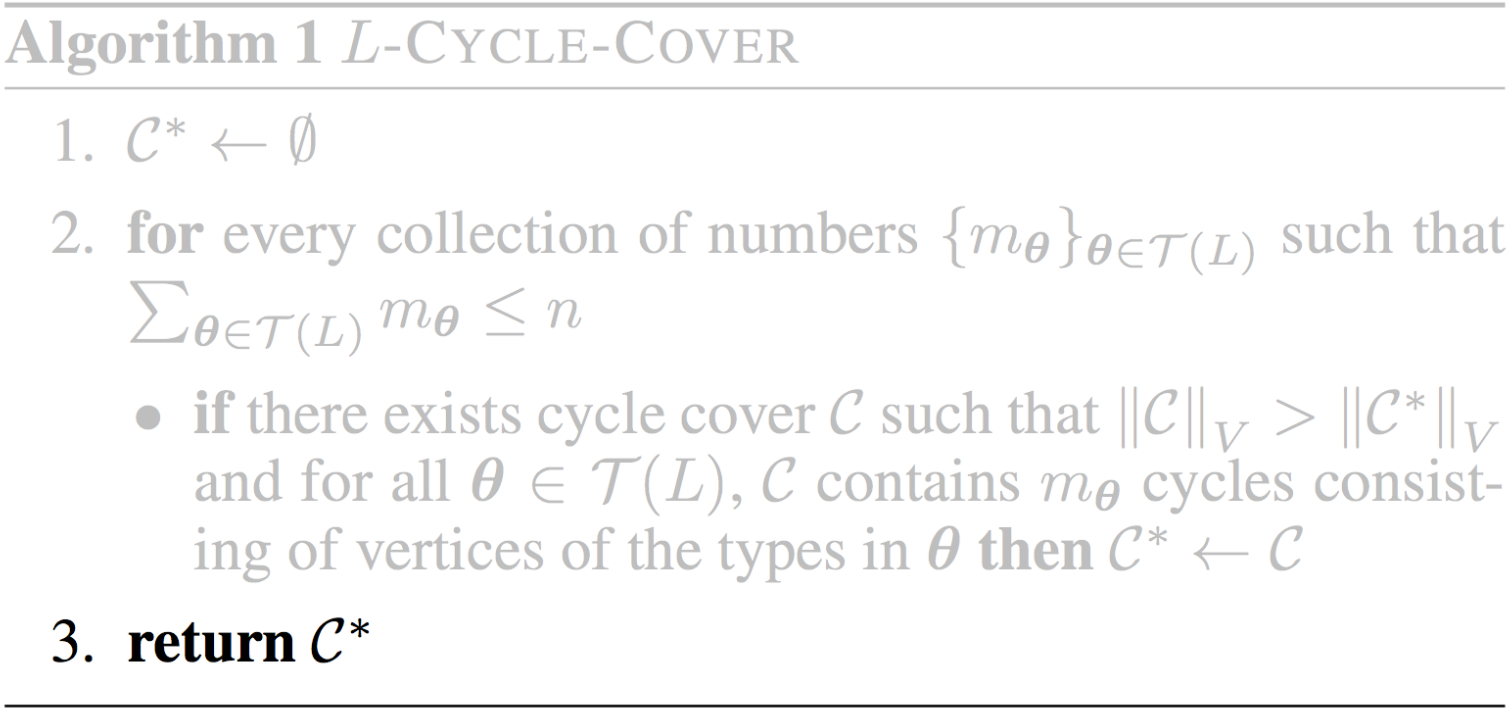


Algorithm 1 L-CYCLE-COVER

```
$\mathcal{C}^* \leftarrow \emptyset$
for every collection of numbers $\{m_{\boldsymbol{\theta}}\}_{\boldsymbol{\theta} \in \mathcal{T}(L)}$ such that $\sum_{\boldsymbol{\theta} \in \mathcal{T}(L)} m_{\boldsymbol{\theta}} \leq n$
    • if there exists cycle cover $\mathcal{C}$ such that $\|\mathcal{C}\|_V > \|\mathcal{C}^*\|_V$ and for all $\boldsymbol{\theta} \in \mathcal{T}(L)$, $\mathcal{C}$ contains $m_{\boldsymbol{\theta}}$ cycles consisting of vertices of the types in $\boldsymbol{\theta}$ then $\mathcal{C}^* \leftarrow \mathcal{C}$
return $\mathcal{C}^*$
```

CLEARING IS NOW IN POLYNOMIAL TIME

- **Return the legal cycle cover such that the sum over θ of m_θ is maximized – aka the largest legal cycle cover**



Algorithm 1 L-CYCLE-COVER

```
$\mathcal{C}^* \leftarrow \emptyset$
**for** every collection of numbers $\{m_\theta\}_{\theta \in \mathcal{T}(L)}$ such that $\sum_{\theta \in \mathcal{T}(L)} m_\theta \leq n$
    • **if** there exists cycle cover $\mathcal{C}$ such that $\|\mathcal{C}\|_V > \|\mathcal{C}^*\|_V$ and for all $\theta \in \mathcal{T}(L)$, $\mathcal{C}$ contains $m_\theta$ cycles consisting of vertices of the types in $\theta$ **then** $\mathcal{C}^* \leftarrow \mathcal{C}$
**return** $\mathcal{C}^*$
```

FLIPPING ATTRIBUTES IS ALSO EASY

- **The human body tries to reject transplanted organs**
  - Before transplantation, can immunnosupress some "bad" traits of the patient to increase transplant opportunity
  - Takes a toll on the patient's health
- **Suppose we can pay some cost to change attributes**
- **For all θ, θ' in Θ, let**

  c : Θ x Θ → R be cost of flipping θ → θ'
- **Flip-and-Cover: maximize match size minus cost of flips**

THEOREM

Given constant *L* and |Θ|,
the Flip-and-Cover problem is in polynomial time

A CONCRETE INSTANTIATION: THRESHOLDING

- **Associate with each patient and donor a *k*-bit vector**
  - Count “conflict bits” that overlap at same position
  - If more than threshold *t* conflict bits, don’t draw an edge
- **Example: *k* = 2, blood containing antigens A and B**
  - $\Theta = 2^{\underbrace{\{\text{ has-A, has-B }\}}_{\text{Donor blood type}}} \times 2^{\underbrace{\{\text{ no-A, no-B }\}}_{\text{Patient blood type}}}$

Donor type A = [1, 0]
Patient type AB = [0, 0] ✓

Donor type A = [1, 0]
Patient type O = [1, 1] ✗

- **Draw edge if $\langle d_i, p_j \rangle \leq t$; do not draw edge otherwise**

ASIDE

Related to **intersection graphs**:
Each vertex has a set; draw edge between vertices iff sets intersect (by at least *p* elements)

UPPER BOUND: SOMETIMES YOU NEED LOTS OF BITS

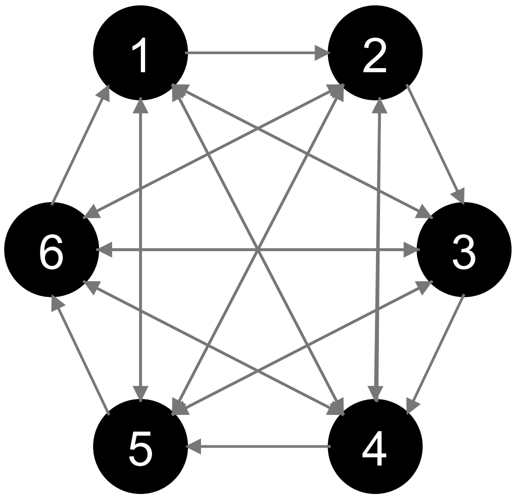
THEOREM

For any $n > 2$, there exists a graph on n vertices that is not $(k,0)$-representable for all $k < n$

For each vertex *i*, draw edge to each vertex except vertices *i*-1 and *i*

BWOC assume (*k*,0)-representable, *k* < *n*:

- **Consider vertex 1**
- **(1, *n*) not in *E*; (1, *i*) in *E* otherwise**
- **Then there is a conflict bit between vertex 1 and *n* that is not "turned on" anywhere else**
- **Do for *n* vertices → require *k* ≥ *n***



HARDNESS: HOW MANY BITS DO I NEED FOR THIS GRAPH?

Given: an input graph $G = (V, E)$
subset F of C(V, 2)

fixed positive k, nonnegative t

Does there exist:

k-length bit vectors d_i, p_i for all v_i in V

such that for (i,j) in F, also (i,j) in E iff $<d_i,p_j> \leq t$

THEOREM

The (k,t)-representation problem is NP-complete

(proof via reduction from 3SAT)

COMPUTING (*K*, *T*)-REPRESENTATIONS: QCP

If an edge does not exist, make sure the overlap is greater than *t*

If an edge exists in the graph, assert the source donor vector and sink patient

- **Quadratically-constrained discrete feasibility program:**
  - Constraint matrix not positive semi-definite → non-convex
- **State-of-the-art nonlinear solvers (e.g., Bonmin) fail**

[Bonami et al. 2008]

COMPUTING (*K*, *T*)-REPRESENTATIONS: IP

$$
\begin{array}{lrr}
\min & \sum_{v_i \in V} \sum_{v_j \neq v_i \in V} \xi_{ij} & \\
\text{s.t.} & d_i^q \geq c_{ij}^q \wedge p_j^q \geq c_{ij}^q & \forall v_i \neq v_j \in V, q \in [k] \\
& d_i^q + p_j^q \leq 1 + c_{ij}^q & \forall v_i \neq v_j \in V, q \in [k] \\
& \sum_q c_{ij}^q \leq t + (k-t)\xi_{ij} & \forall (v_i, v_j) \in E \\
& \sum_q c_{ij}^q \geq (t+1)\xi_{ij} & \forall (v_i, v_j) \in E \\
& \sum_q c_{ij}^q \geq t + 1 - k\xi_{ij} & \forall (v_i, v_j) \notin E \\
& \sum_q c_{ij}^q \leq k - (k-t)\xi_{ij} & \forall (v_i, v_j) \notin E \\
& d_i^q, p_i^q \in \{0,1\} & \forall v_i \in V, q \in [k] \\
& c_{ij}^q, \xi_{ij} \in \{0,1\} & \forall v_i \neq v_j \in V, q \in [k]
\end{array}
$$

- **Integer program minimizes number of “conflict edges”**
  - CPLEX struggles to find non-trivial solutions
  - CPLEX cannot find feasible solution (when forcing all ξ_{ij} = 0)

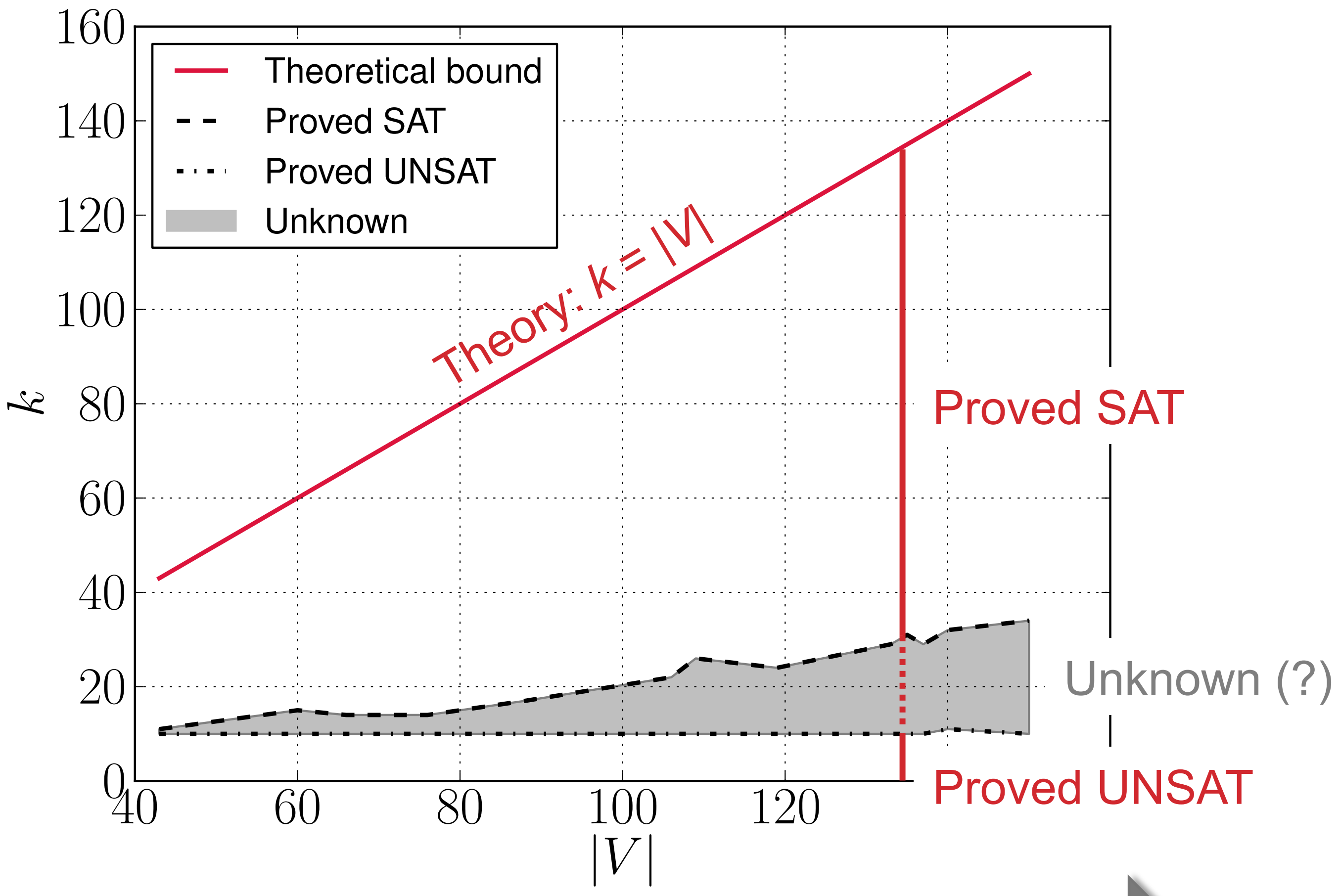
COMPUTING (*K*,*0*)-REPRESENTATIONS: SAT

Specific case of $t = 0$: if an edge does not exist, force any overlap

Specific case of $t = 0$: if an edge exists, allow no overlap

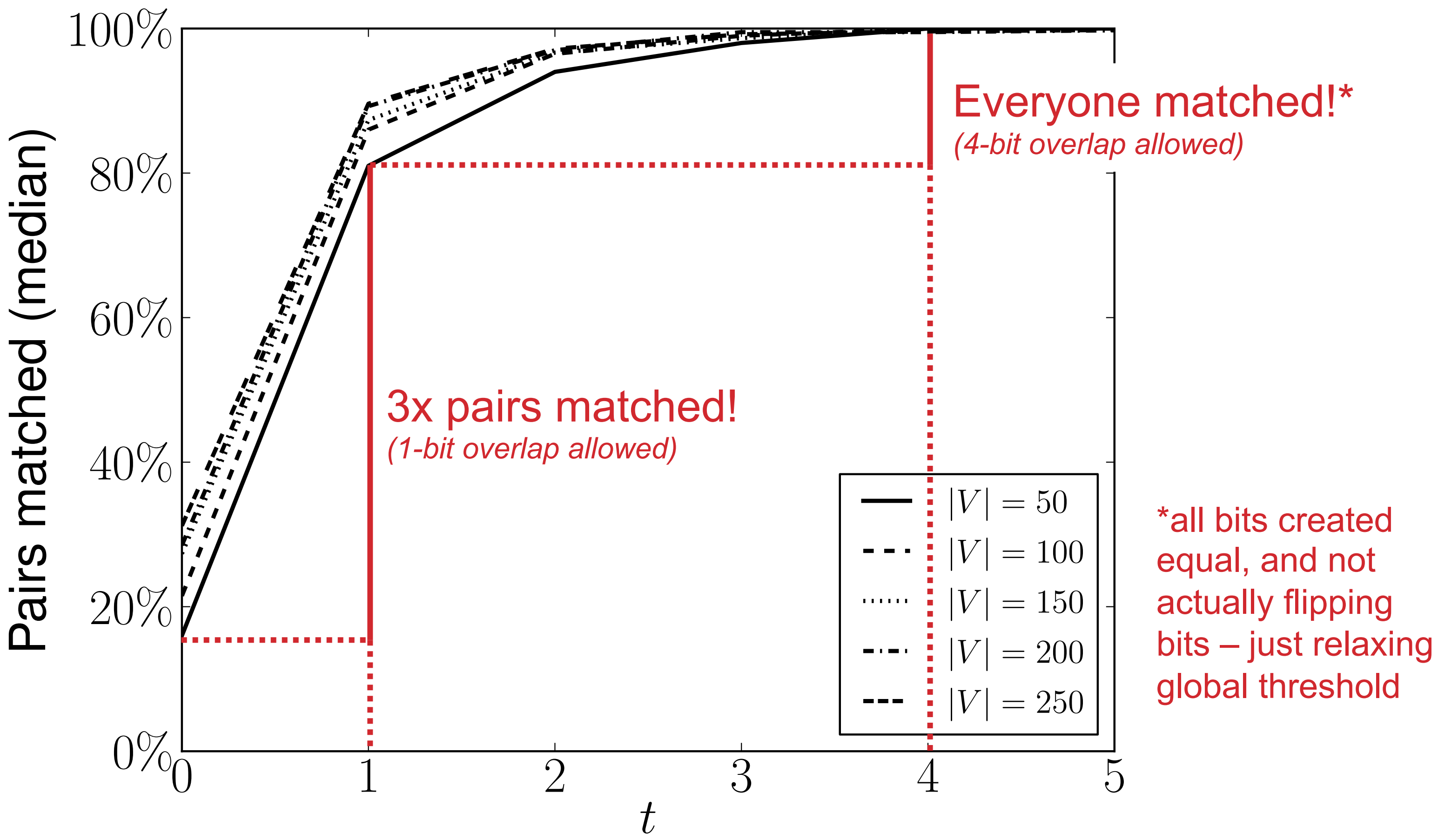
- **When $t = 0$, can use a compact SAT formulation**
  - Interesting because it closely mimics real life
- **We can solve small- and medium-sized graphs**
  - Use Lingeling, a good parallel SAT solver [Biere 2014]

CAN WE REPRESENT REAL GRAPHS WITH A SMALL NUMBER OF BITS?



Bigger real-world graphs (UNOS 2010 – 2012)

RELAXING THE THRESHOLD



3x pairs matched!

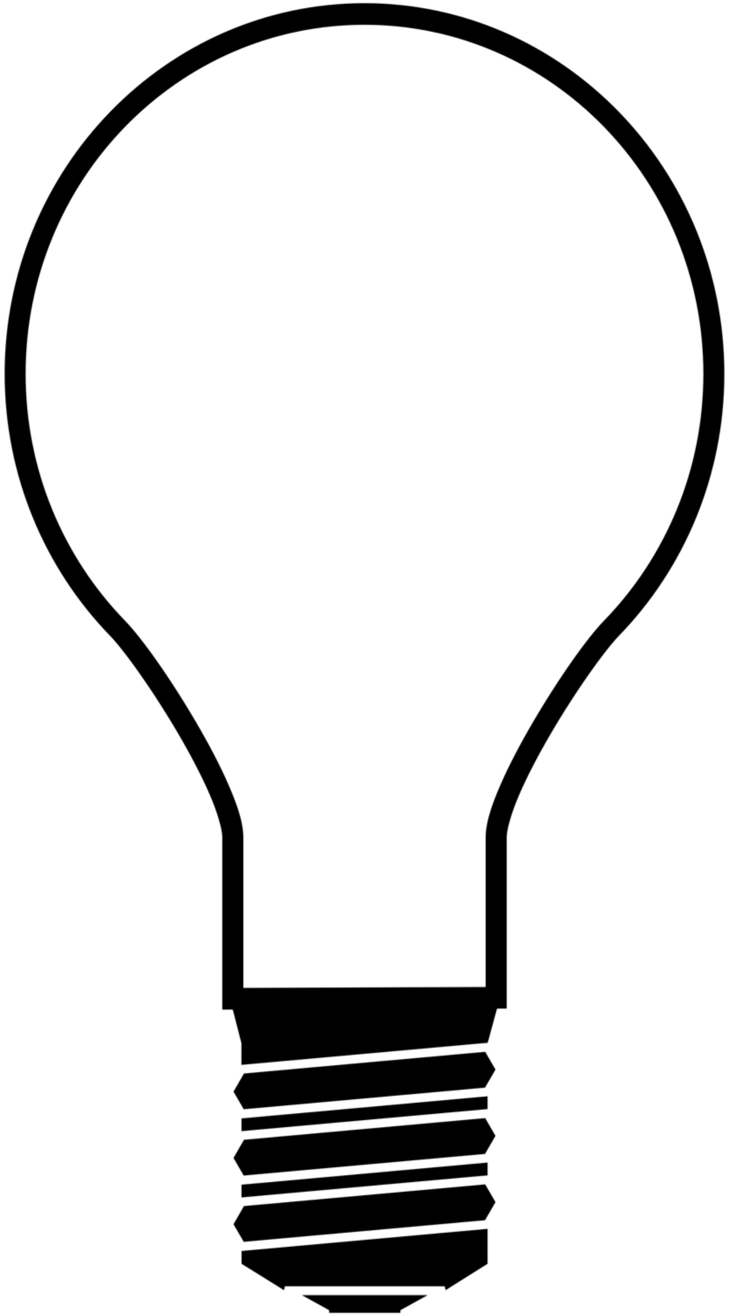
(1-bit overlap allowed)

Everyone matched!*

(4-bit overlap allowed)

*all bits created equal, and not actually flipping bits – just relaxing global threshold

Loosen bit threshold *t* on real UNOS graphs



BACKUP SLIDES

JOHN P DICKERSON

FAILURE-AWARE MODEL

Compatibility graph *G*

- Edge (v_i, v_j) if v_i's donor can donate to v_j's patient
- Weight w_e on each edge *e*

Success probability q_e for each edge *e*

Discounted utility of cycle *c*

$$u(c) = \sum w_e \bullet \prod q_e$$

Value of successful cycle

Probability of success

FAILURE-AWARE MODEL

Discounted utility of a *k*-chain *c*

$$u(c) = \left[\sum_{i=1}^{k-1}(1-q_i)i\prod_{j=0}^{i-1} q_j\right] + \left[k\prod_{i=0}^{k-1} q_i\right]$$

Exactly first *i* transplants

Chain executes in entirety

Cannot simply "reweight by failure probability"

Utility of a match *M*: ***u*(*M*) = ∑ *u*(*c*)**

INCREMENTALLY SOLVING VERY LARGE IPS

#Decision variables grows linearly with #cycles and #chains in the pool

- Millions, billions of variables
- Too large to fit in memory

Branch-and-price incrementally brings variables into a reduced model [Barnhart et al. 1998]

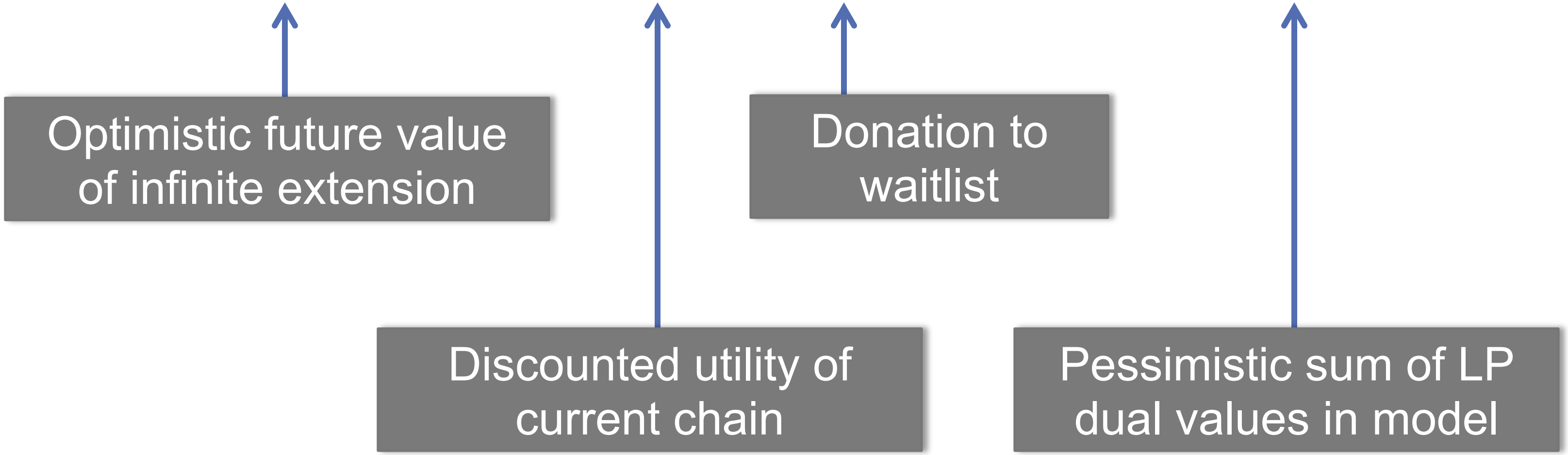
Solves the "pricing problem" – each variable gets a real-valued price

- Positive price → resp. constraint in full model violated
- No positive price cycles → optimality at this node

CONSIDERING ONLY “GOOD” CHAINS

Theorem:
Given a chain *c*, any extension *c’* will not be needed in an optimal solution if the infinite extension has non-positive value.

$$\left(\frac{q_{max}}{1-q_{max}}\prod_{i=0}^{k-1} q_i\right) + u(c) + \ell - \left(d_{min} + \sum_{i=0}^{k} d_i\right) \leq 0$$



Optimistic future value of infinite extension

Donation to waitlist

Discounted utility of current chain

Pessimistic sum of LP dual values in model

$G(n, t(n), p)$: random graph with

- n patient-donor pairs
- $t(n)$ altruistic donors
- Probability $\Theta(1/n)$ of incoming edges

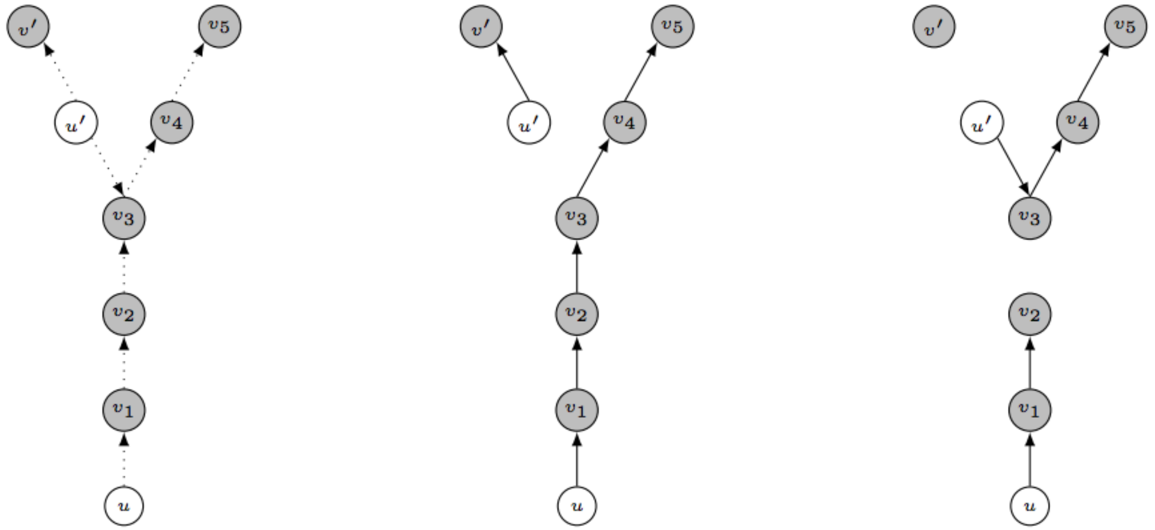
Constant transplant success probability q

Theorem

For all $q \in (0,1)$ and $\alpha, \beta > 0$, given a large $G(n, \alpha n, \beta/n)$, w.h.p. there exists some matching M' s.t. for every maximum cardinality matching M,

$$u_q(M') \geq u_q(M) + \Omega(n)$$

BRIEF INTUITION: COUNTING Y-GADGETS



(a) A Y gadget.

(b) The maximum cardinality matching M_Y.

(c) The matching M'_Y.

For every structure *X* of constant size, w.h.p. can find *Ω*(*n*) structures isomorphic to *X and isolated from the rest of the graph*

Label them (alt vs. pair): flip weighted coins, constant fraction are labeled correctly → constant × *Ω*(*n*) = *Ω*(*n*)

Direct the edges: flip 50/50 coins, constant fraction are entirely directed correctly → constant × *Ω*(*n*) = *Ω*(*n*)

Under the “most stringent” fairness rule:

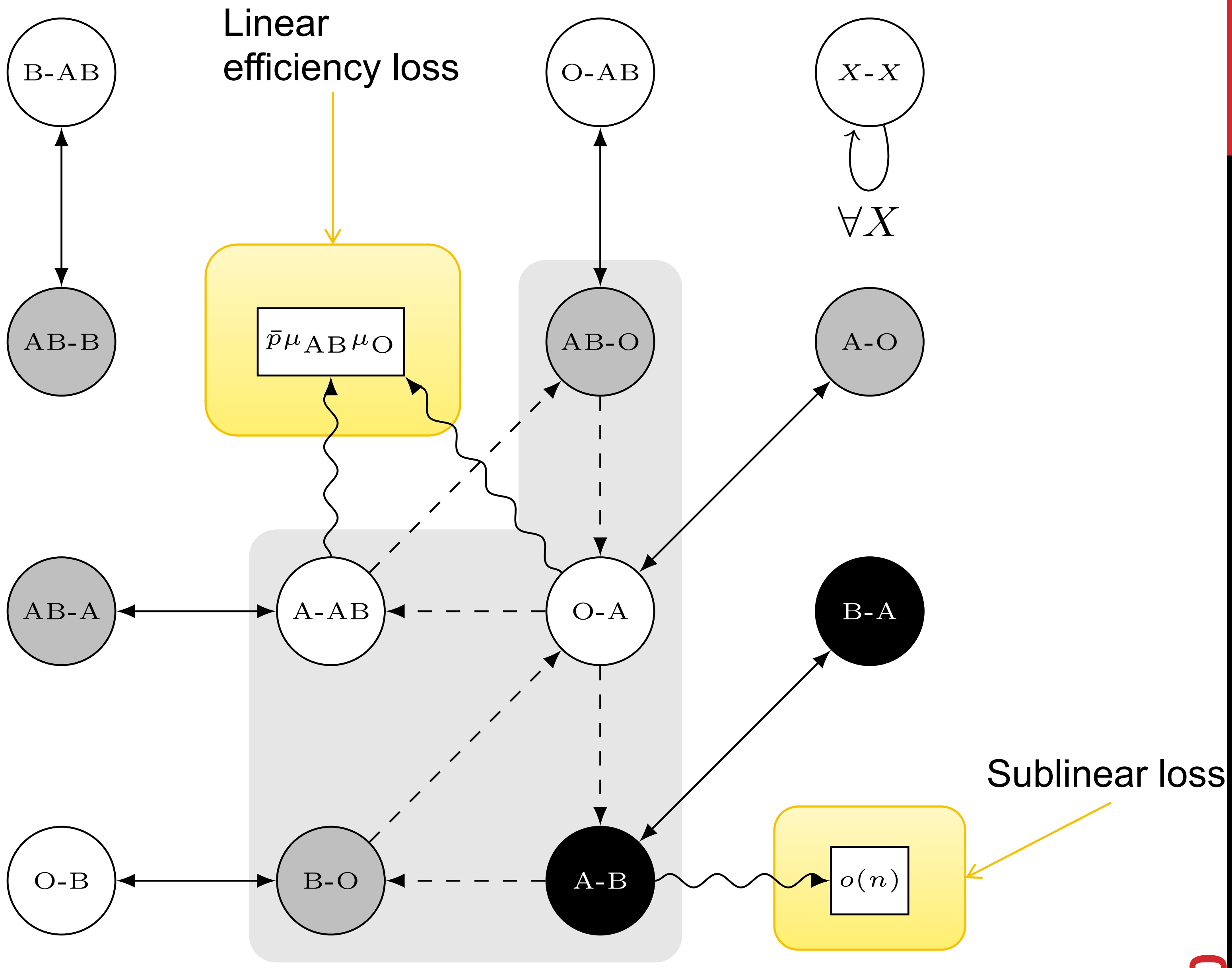
$$u_{H \succ L}(M) = \begin{cases} u(M) & \text{if } |M_H| = \max_{M' \in \mathcal{M}} |M'_H| \\ 0 & \text{otherwise} \end{cases}$$

Theorem

Assume “reasonable” level of sensitization and “reasonable” distribution of blood types. Then, almost surely as *n* → ∞,

$$\text{POF}(\mathcal{M}, u_{H \succ L}) \leq \frac{2}{33}.$$

(And this is achieved using cycles of length at most 3.)



Linear efficiency loss

Sublinear loss

BETTER STATIC OPTIMIZATION METHODS

Recall two main methods for solving big IPs for kidney exchange:

- Branch-and-price = B&B + column generation
- Constraint generation

Many different ways to do these:

- E.g., how do I solve the pricing problem?
- E.g., which constraints should I add to the model?

Big runtime changes [Anderson et al. PNAS-2015, Glorie et al. MSOM-2014]

BASIC EDGE FORMULATION

[Abraham et al. 07]

Binary variable x_{ij} for each edge from *i* to *j*

Maximize

$$u(M) = \Sigma\, \mathrm{w}_{\mathrm{ij}}\, x_{ij}$$

Flow constraint

Subject to

$\Sigma_{\mathrm{j}}\, x_{ij} = \Sigma_{\mathrm{j}}\, x_{ji}$ for each vertex *i*

$\Sigma_{\mathrm{j}}\, x_{ij} \leq 1$ for each vertex *i*

$\Sigma_{1\leq k\leq L}\, \mathrm{x}_{\mathrm{i(k)i(k+1)}} \leq \mathrm{L}-1$ for paths i(1)…i(L+1)

(no path of length L that doesn't end where it started – cycle cap)

STATE OF THE ART FOR EDGE FORMULATION

[Anderson et al. PNAS-2015]

Builds on the prize-collecting traveling salesperson problem [Balas Networks-89]

- PC-TSP: visit each city (patient-donor pair) exactly once, but with the additional option to pay some penalty to skip a city (penalized for leaving pairs unmatched)

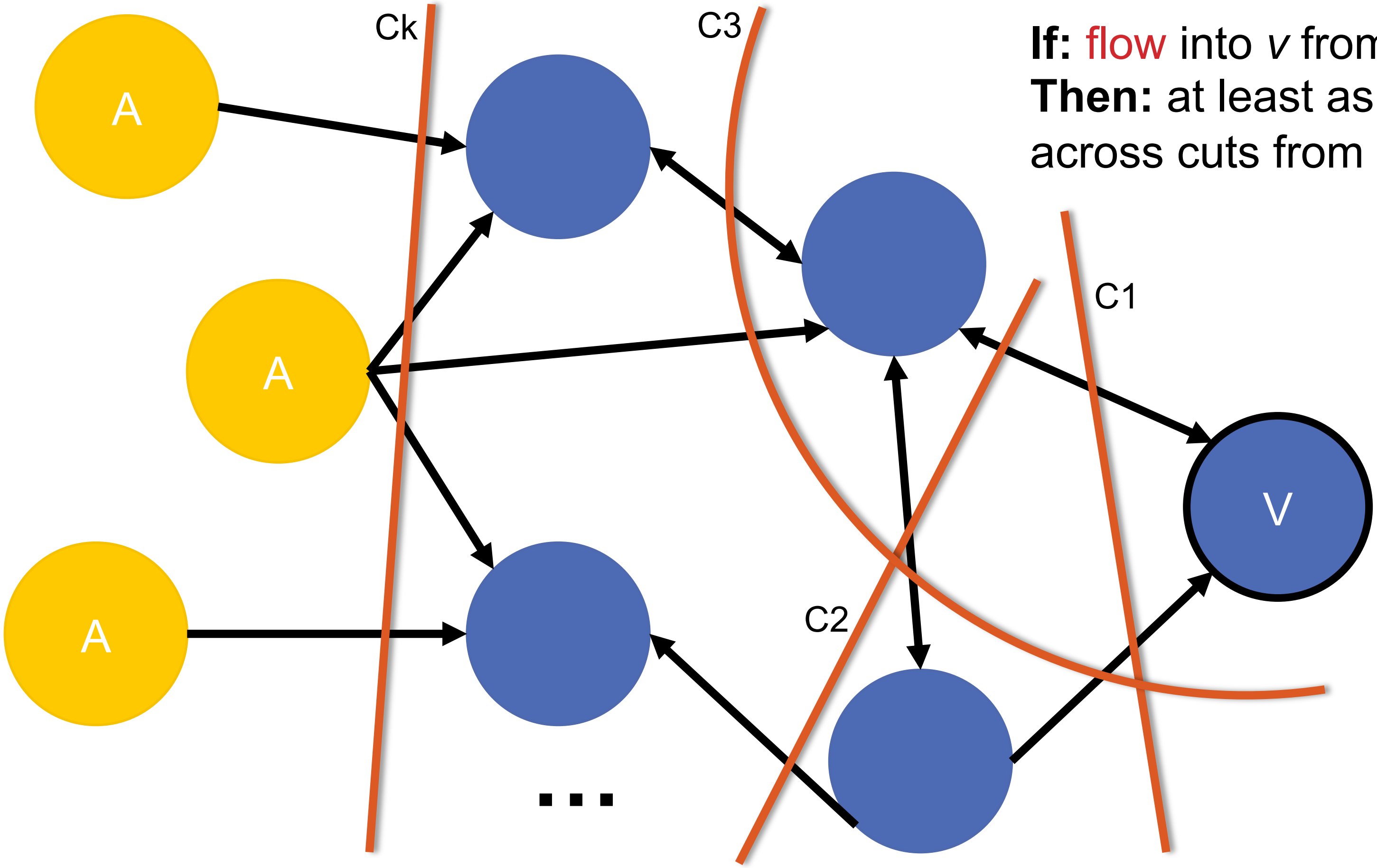
They maintain decision variables for all cycles of length at most *L*, but build chains in the final solution from decision variables associated with individual edges

Then, an exponential number of constraints could be required to prevent the solver from including chains of length greater than *K*; these are generated incrementally until optimality is proved.

- Leverage cut generation from PC-TSP literature to provide stronger (i.e. tighter) IP formulation

BEST EDGE FORMULATION

[Anderson et al. 15]



If: flow into *v* from a chain
Then: at least as much flow across cuts from {A}

REVIEW: CYCLE FORMULATION

Objective = maximum cardinality

Binary variable x_c for each cycle/chain c of length at most L

Maximize

$$\Sigma\ |c| x_c$$

Subject to

$$\Sigma_{c\ :\ i \text{ in } c}\ x_c \leq 1 \quad \text{for each vertex } i$$

DFS TO SOLVE PRICING PROBLEM *[Abraham et al. PNAS-2015]*

Pricing problem:

- Optimal dual solution π^* to reduced model
- Find non-basic variables with **positive price** (for a maximization problem)
  - 0 < weight of cycle – sum of duals in π^* of constituent vertices

First approach [Abraham et al. EC-2007] *explicitly* prices all feasible cycles and chains through a DFS

- Can speed this up in various ways, but proving **no positive price cycles exist** still takes time poly in chain/cycle cap = bad for even reasonable caps

THE RIGHT IDEA

Idea: solve structured optimization problem that implicitly prices variables

Price: $w_c - \Sigma_{v \text{ in } c}\, \delta_v \quad = \quad \Sigma_{e \text{ in } c}\, w_e - \Sigma_{v \text{ in } c}\, \delta_v \quad = \quad \Sigma_{(u,v) \text{ in } c}\, [w_{(u,v)} - \delta_v]$

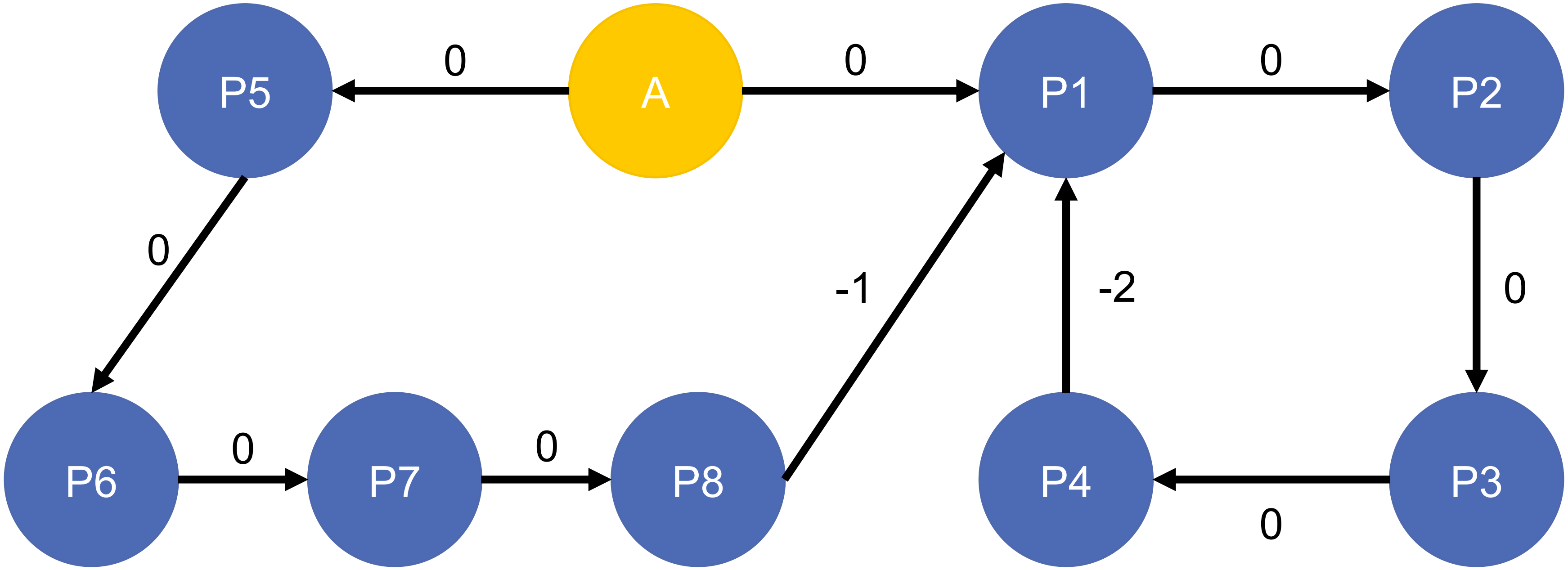
Take *G*, create *G'* s.t. all edges *e* = (*u*,*v*) are reweighted $r_{(u,v)} = \delta_v - w_{(u,v)}$

- Positive price cycles in *G* = negative weight cycles in *G'*

Bellman-Ford finds shortest paths

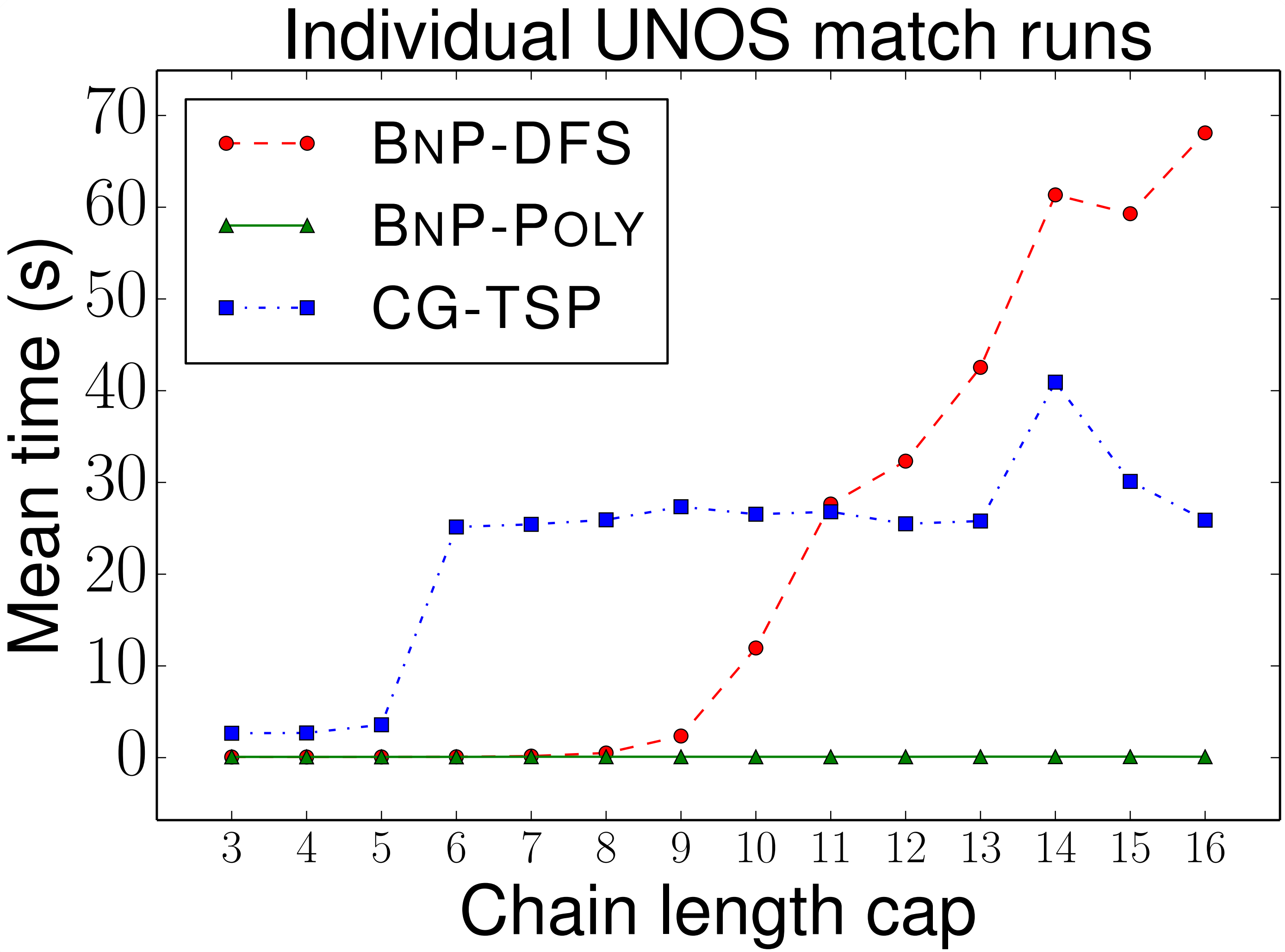
- Undefined in graphs with negative weight
- Adapt B-F to prevent internal looping during the traversal
  - *Shortest* path is NP-hard (reduce from Hamiltonian path:
    - Set edge weights to -1, given edge (*u*,*v*) in *E*, ask if shortest path from *u* to *v* is weight *1*-|*V*| → visits each vertex exactly once
  - We only need *some* short path (or proof that no negative cycle exists)
- Now pricing runs in time $O(|V||E|\text{cap}^2)$

LOOP BLOCKING MUST BE DURING TRAVERSAL



(cycle cap = 3, chain cap = 6)

EXPERIMENTAL RESULTS



Note: Anderson et al.’s algorithm (CG-TSP) is *very strong* for uncapped aka “infinite-length” chains, but a chain cap is often imposed in practice