APPLIED MECHANISM DESIGN FOR SOCIAL GOOD

JOHN P DICKERSON

Lecture #11 – 10/4/2016

CMSC828M
Tuesdays & Thursdays
12:30pm – 1:45pm

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND
THIS CLASS:
BATCH CLEARING OF ORGAN EXCHANGES
THE CLEARING PROBLEM

The clearing problem is to find the “best” disjoint set of cycles of length at most $L$, and chains (maybe with a cap $K$)

• This class: only consider static matching in the present
• Next class: more general dynamic matching over time
SPECIAL CASE: $L = 2$

PTIME: translate to maximum matching on undirected graph

(Six pairs, no altruists.)
SPECIAL CASE: $L = \infty$

PTIME via formulation as maximum weight perfect matching

(Six pairs, no altruists.)

Donors: $d_1, d_2, d_3, d_4, d_5, d_6$

Patients: $p_1, p_2, p_3, p_4, p_5, p_6$

Edge weights:

- $\cdots = 0$
- $\cdot\cdot\cdot = w_e$
GENERAL CASE: $L = ?$

NP-hard via reduction from 3D-matching:

- Given disjoint sets $X, Y, Z$ of size $q$ ...
- ... and a set of triples $T \subseteq X \times Y \times Z$ ...
- ... is there a disjoint subset $M \subseteq T$ 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 $X \cup Y \cup Z$
GENERAL CASE: $L = ?$

$M$ is perfect matching $\rightarrow$ construction has perfect cycle cover.

For $t_i$ in $T$:
GENERAL CASE: \( L = ? \)

M is perfect matching \( \rightarrow \) construction has perfect cycle cover.

For \( t_i \) not in \( T \):

\[
\begin{align*}
X_a^i & \rightarrow L-1 \\
Y_b^i & \rightarrow L-1 \\
Z_c^i & \rightarrow L-1
\end{align*}
\]
GENERAL CASE: \( L = ? \)

We have a perfect cycle cover \( \rightarrow M \) 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

$$u(M) = \sum w_{ij} x_{ij}$$

Subject to

$$\sum_j x_{ij} = \sum_j x_{ji}$$

for each vertex $i$

$$\sum_j x_{ij} \leq 1$$

for each vertex $i$

$$\sum_{1 \leq k \leq L} x_{i(k) i(k+1)} \leq L - 1$$

for paths $i(1) \ldots i(L+1)$

(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
If: flow into $v$ from a chain
Then: at least as much flow across cuts from \{A\}
Binary variable $x_c$ for each feasible cycle or chain $c$

Maximize

$$u(M) = \sum w_c x_c$$

Subject to

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

```
x_7
  1  0
```

- 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)
- $OPT(P') \leq OPT(P)$

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 $\pi^*$ to reduced model
- Find non-basic variables with positive price (for a maximization problem)
  - $0 < \text{weight of cycle} - \text{sum of duals in } \pi^* \text{ of constituent vertices}$
  - Positive price for cycle $\rightarrow$ dual constraint is violated
  - No positive price cycles $\rightarrow$ 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

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 $\frac{1}{2}$ on each edge


- 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

maximize \[ \sum_{f} \sum_{(i,j) \in A} w_{ij} x_{ij}^f \]
subject to \[ \sum_{j: (j,i) \in A} x_{ij}^f = \sum_{j: (i,j) \in A} x_{ij}^f \quad \forall i \in V, \forall f \in \{1, \ldots, F\} \]
\[ \sum_{f} \sum_{j: (i,j) \in A} x_{ij}^f \leq 1 \quad \forall i \in V \]
\[ \sum_{(i,j) \in A} x_{ij}^f \leq k \quad \forall f \in \{1, \ldots, F\} \]
\[ x_{ij}^f \in \{0, 1\} \quad \forall (i,j) \in A, \forall f \in \{1, \ldots, F\} \]

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

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

[Dickerson et al. EC-16]

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

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**

[Dickerson et al. EC-16]

Maximize

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

Subject to

\[ \sum_{ij \in E} \sum_{k \in K'(i,j)} y_{ijk} + \sum_{c : i \in c} z_c \leq 1 \quad \text{for every } i \text{ in Pairs} \]

*Each pair can be in at most one chain or cycle*

\[ \sum_{ij \in E} y_{ij1} \leq 1 \quad \text{for every } i \text{ in Altruists} \]

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

\[ \sum_{j : ij \in E} y_{ijk+1} - \sum_{j : ji \in E} \sum_{k \in K'(j,i)} y_{jik} \leq 0 \quad \text{for every } i \text{ in Pairs and } k \in \{1, \ldots, K-1\} \]

*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?

[Dickerson et al. EC-16]

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!

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 \( x_c = w_c - \sum_{v \in c} \delta_v \)

Example

- Price: \((2+3+2) - (\delta_{P1} + \delta_{P2} + \delta_{P3})\)

\[
\begin{align*}
&= \sum_{e \in c} w_e - \sum_{v \in c} \delta_v \\
&= \sum_{(u,v) \in c} [w_{(u,v)} - \delta_v]
\end{align*}
\]

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 \(O(|V||E|L^2)\)
FAILURE-AWARE KIDNEY EXCHANGE
[Dickerson et al. EC-13, EC-16]

In practice, not all edges exist; lots of recent work [Li et al. 2011, Dickerson et al. 2013, Blum et al. 2013, Anderson et al. 2015, Blum et al. 2015, Glorie et al. 2016, Pedroso&Ikeda 2016, Assadi et al. 2016]

One approach: associate a success probability $p$ with each edge, maximize expected size of remaining matching after independent edge failures [Dickerson et al. 2013]:

- Cycles succeed only if all edges succeed
- Chains succeed up to first edge failure

Earlier compact formulations cannot be adapted to this model due to expected utility of edge changing based on position

Minor adjustment to PICEF’s objective function:

$$u_p(M) = \sum_{ij \in E} \sum_{k \in \kappa'(i,j)} p^k w^{ij} y_{ijk} + \sum_{c \in C} p^{|c|} w_c z_c$$

Can also adapt Bellman-Ford to give a failure-aware polynomial time pricing algorithm for cycles

More on uncertainty next lecture!
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

Individual UNOS match runs

Mean time (s)

Chain length cap

NLDKSS: 17 match runs

Individual NLDKSS match runs

Mean time (s)

Chain length cap
GENERATED DATA

$|P|=700$, INCREASING %ALTRUISTS
Solvers that are not shown timed out (within one-hour period).
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

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 $q$th 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 $\Theta$ 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 $\rightarrow$ chains are now cycles)
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 $\theta$ to a patient with type $\theta'$
- For all $\theta = (\langle \theta_1, p, \theta_1, d \rangle, \ldots, \langle \theta_r, p, \theta_r, d \rangle)$ in $\Theta^2 r$ 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 $T(L) = \{ \theta \in \Theta^{2r} : r \leq L \text{ and } f_C(\theta) = 1 \}$
CLEARING IS NOW IN POLYNOMIAL TIME

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

\begin{algorithm}
\caption{$L$-Cycle-Cover}

1. $C^* \leftarrow \emptyset$

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

3. return $C^*$
\end{algorithm}
CLEARING IS NOW IN POLYNOMIAL TIME

• Each set \( \{m_\theta\} \) says we have \( m_{\theta_1} \) cycles of type \( \theta_1 \), \( m_{\theta_2} \) cycles of \( \theta_2 \), ..., \( m_{\theta_{|T(L)|}} \) cycles of \( \theta_{|T(L)|} \), constrained to at most \( n \) cycles total

Algorithm 1 \textit{L-Cycle-Cover}

1. \( C^* \leftarrow \emptyset \)

2. \textbf{for} every collection of numbers \( \{m_\theta\}_{\theta \in T(L)} \) such that \( \sum_{\theta \in T(L)} m_\theta \leq n \)

   • if there exists cycle cover \( C \) such that \( \|C\|_V > \|C^*\|_V \) and for all \( \theta \in T(L) \), \( C \) contains \( m_\theta \) cycles consisting of vertices of the types in \( \theta \) then \( C^* \leftarrow C \)

3. \textbf{return} \( C^* \)
CLEARING IS NOW IN POLYNOMIAL TIME

• Check to see if this collection is a legal cycle cover – just check that each type $\theta$ isn’t used too many times in $m_\theta$

Algorithm 1 $L$-Cycle-Cover

1. $C^* \leftarrow \emptyset$

2. 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 $C$ such that $\|C\|_V > \|C^*\|_V$ and for all $\theta \in \mathcal{T}(L)$, $C$ contains $m_\theta$ cycles consisting of vertices of the types in $\theta$ then $C^* \leftarrow C$

3. return $C^*$
CLEARING IS NOW IN POLYNOMIAL TIME

- Return the legal cycle cover such that the sum over $\theta$ of $m_\theta$ is maximized – aka the largest legal cycle cover

Algorithm 1 $L$-Cycle-Cover

1. $C^* \leftarrow \emptyset$

2. 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 $C$ such that $\|C\|_V > \|C^*\|_V$ and for all $\theta \in \mathcal{T}(L)$, $C$ contains $m_\theta$ cycles consisting of vertices of the types in $\theta$ then $C^* \leftarrow C$

3. return $C^*$
FLIPPING ATTRIBUTES IS ALSO EASY

- The human body tries to reject transplanted organs
  - Before transplantation, can immunosuppress 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 $\theta, \theta'$ in $\Theta$, let $c : \Theta \times \Theta \to \mathbb{R}$ be cost of flipping $\theta \to \theta'$
- Flip-and-Cover: maximize match size minus cost of flips

Given constant $L$ and $|\Theta|$, 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
  - $\Theta = 2\{\text{has-A, has-B}\} \times 2\{\text{no-A, no-B}\}$
  - Draw edge if $<d_i, p_j> \leq t$; do not draw edge otherwise

### Related to intersection graphs:
Each vertex has a set; draw edge between vertices iff sets intersect (by at least $p$ elements)
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 \( \rightarrow \) require \( k \geq n \)
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 $<d_i, p_j> \leq t$

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 $\rightarrow$ non-convex
- State-of-the-art nonlinear solvers (e.g., Bonmin) fail

[Bonami et al. 2008]
COMPUTING \((K,T)\)-REPRESENTATIONS: IP

\[
\begin{aligned}
\min & \quad \sum_{v_i \in V} \sum_{v_j \neq v_i \in V} \xi_{ij} \\
\text{s.t.} \quad & d^q_i \geq c^q_{ij} \land p^q_j \geq c^q_{ij} \\
& d^q_i + p^q_j \leq 1 + c^q_{ij} \quad \forall v_i \neq v_j \in V, q \in [k] \\
& \sum_q c^q_{ij} \leq t + (k - t)\xi_{ij} \quad \forall (v_i, v_j) \in E \\
& \sum_q c^q_{ij} \geq (t + 1)\xi_{ij} \quad \forall (v_i, v_j) \in E \\
& \sum_q c^q_{ij} \geq t + 1 - k\xi_{ij} \quad \forall (v_i, v_j) \notin E \\
& \sum_q c^q_{ij} \leq k - (k - t)\xi_{ij} \quad \forall (v_i, v_j) \notin E \\
& d^q_i, p^q_j \in \{0, 1\} \quad \forall v_i \in V, q \in [k] \\
& c^q_{ij}, \xi_{ij} \in \{0, 1\} \quad \forall v_i \neq v_j \in V, q \in [k]
\end{aligned}
\]

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

Theoretical bound
Proved SAT
Proved UNSAT
Unknown

Bigger real-world graphs (UNOS 2010 – 2012)
RELAXING THE THRESHOLD

Loosen bit threshold $t$ on real UNOS graphs

Everyone matched!*
(4-bit overlap allowed)

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

3x pairs matched!
(1-bit overlap allowed)
NEXT CLASS:
DYNAMIC OPTIMIZATION