

# Applied Discrete Optimization (WI000819)

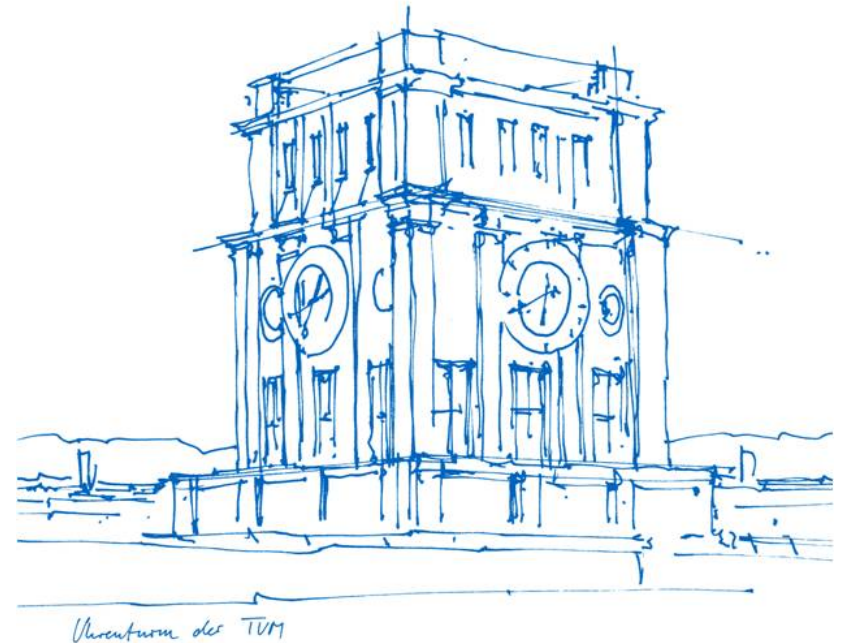
## Kidney Exchange Problem

Prof. Dr. Andreas S. Schulz

Technische Universität München

TUM School of Management and TUM School of Computation, Information and Technology

Operations Research Group



# Kidney failure is a huge part of American health care, and one of the defining qualifiers for Medicare

Medicare is health insurance for the following

- People 65 or older
- People under 65 with certain disabilities
- **People of any age with End-Stage Renal Disease (ESRD) (permanent kidney failure requiring dialysis or a kidney transplant)**

# Budgetary Cost of Kidney Failure

- CMS (medicare) spends \$35B/year on ESRD (7.2% of total budget)
- Other payers spend \$15B/year on ESRD

ESRD costs US healthcare system \$50B/year  
1.6% of all US healthcare spending

Market design innovations to change the way kidney transplants are done will involve dealing with many existing institutions and interests. So incentives are involved, as was as optimization...

# Kidney transplants save money

- Dialysis costs \$89,000/year
- Kidney transplantation costs \$33,000/year
  - ~\$100,000 for a transplant
  - then immunosuppressive drugs

**In 5 years tax payers save >\$275,000 per kidney transplant**

# Kidney transplants save lives

Patients live 10 years longer if they are transplanted than had they remained on dialysis

# Kidney Exchange Background

- There are more than 119,190 patients on the waiting list for cadaver kidneys in the US according UNOS (United Network for Organ Sharing).
- In 2012, there were 11,579 transplants of cadaver kidneys performed in the U.S and more than 5,771 from living donor.
- About 5,000 patients die per year while on the waiting list. Many others are removed from the list as “Too Sick to Transplant”.

# Kidney Exchange Background

- Family or friends are willing to donate a kidney, but ...
- ... often donors are incompatible with their intended recipients.
- This opens the possibility of *exchange*.

# By law, no money changes hands

Section 301, National Organ Transplant Act (NOTA)

42 U.S.C. 274e 1984:

*“it shall be unlawful for any person to knowingly acquire, receive or otherwise **transfer any human organ for valuable consideration** for use in human transplantation”*





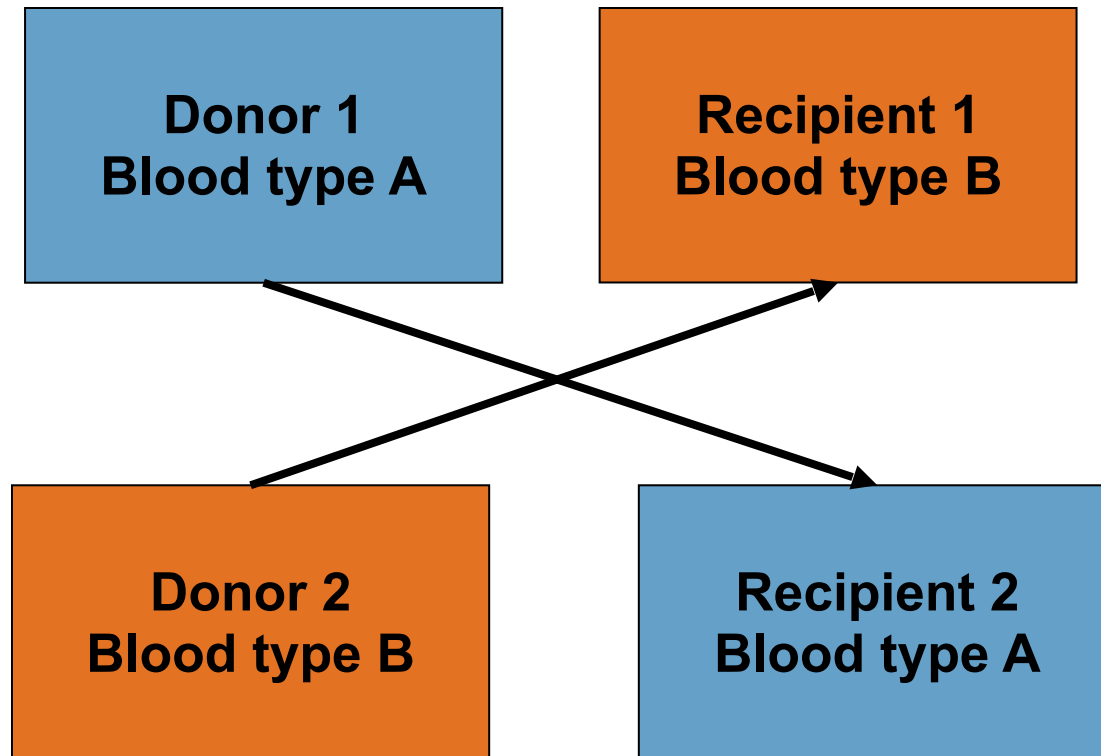
# Charlie W. Norwood Living Organ Donation Act

Public Law 110-144, 110<sup>th</sup> Congress,  
Dec. 21, 2007

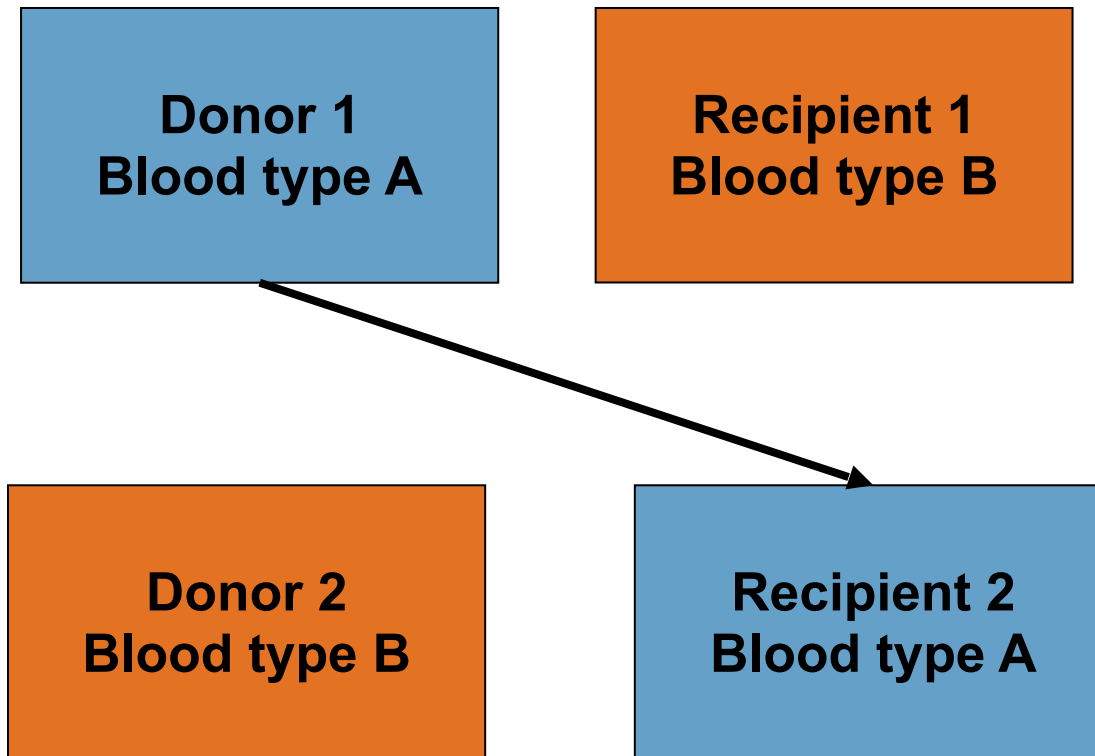
Section 301 of the National Organ Transplant Act (42 U.S.C. 274e) is amended - (1) in subsection (a), by adding at the the end the following:

- “... The preceding sentence **does not** apply with respect to human organ **paired donation**”

# Two pair (2-way) kidney exchange



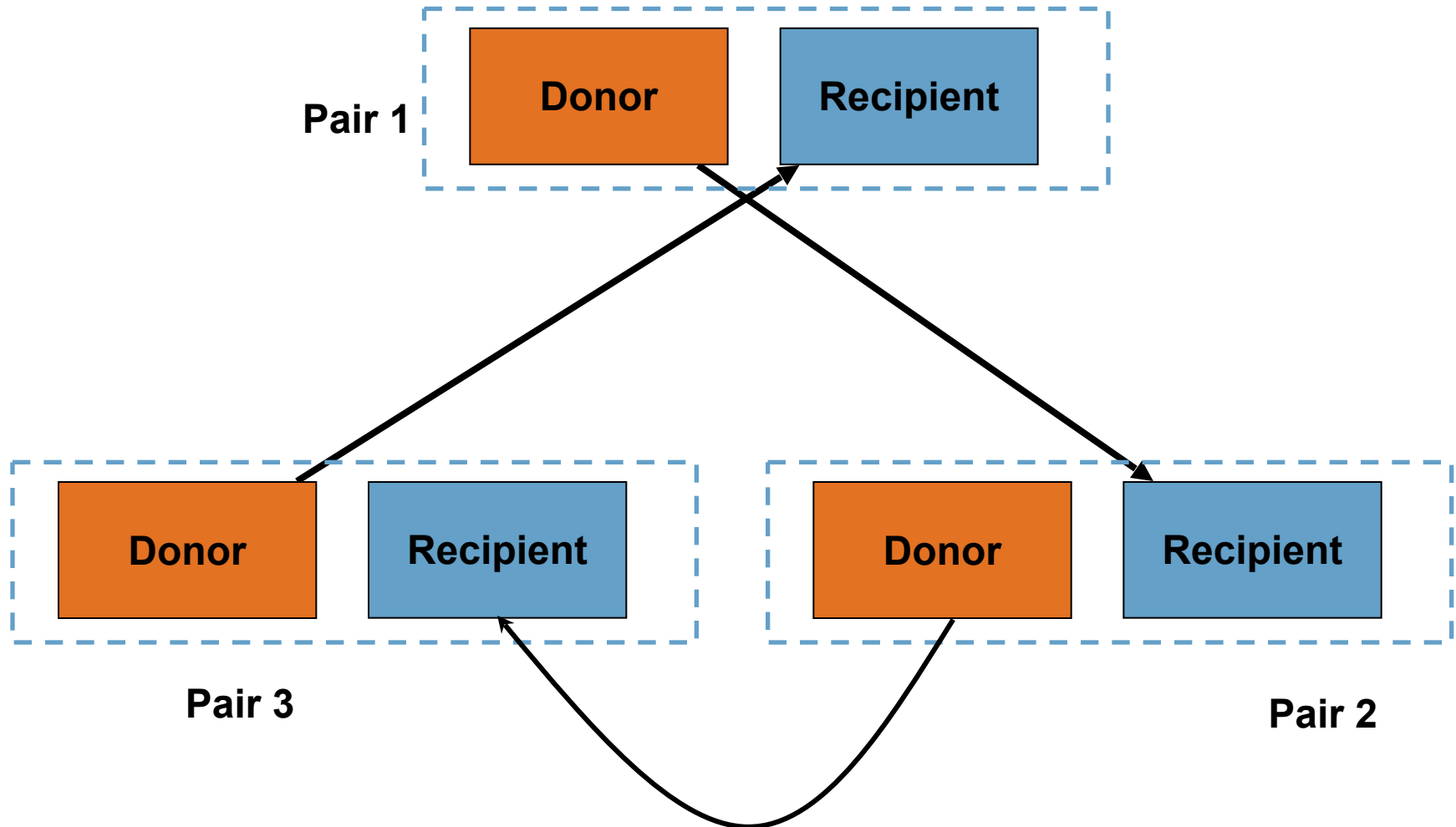
2-way exchange involves 4 *simultaneous* surgeries



## Incentive Constraint: 2-way exchange involves 4 *simultaneous* surgeries



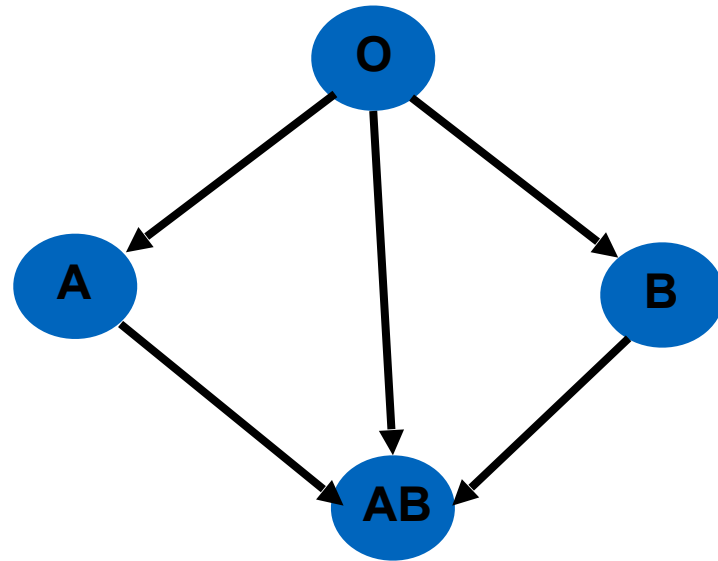
# 3-Way Kidney Exchange





# Factors determining transplant opportunity

- **Blood compatibility**



- **Tissue type compatibility:** Percentage reactive antibodies (PRA)

# Kidney exchange institutions

Alliance Paired Donation (APD) 70+ hospitals

National Kidney Registry (NKR) 75+ hospitals

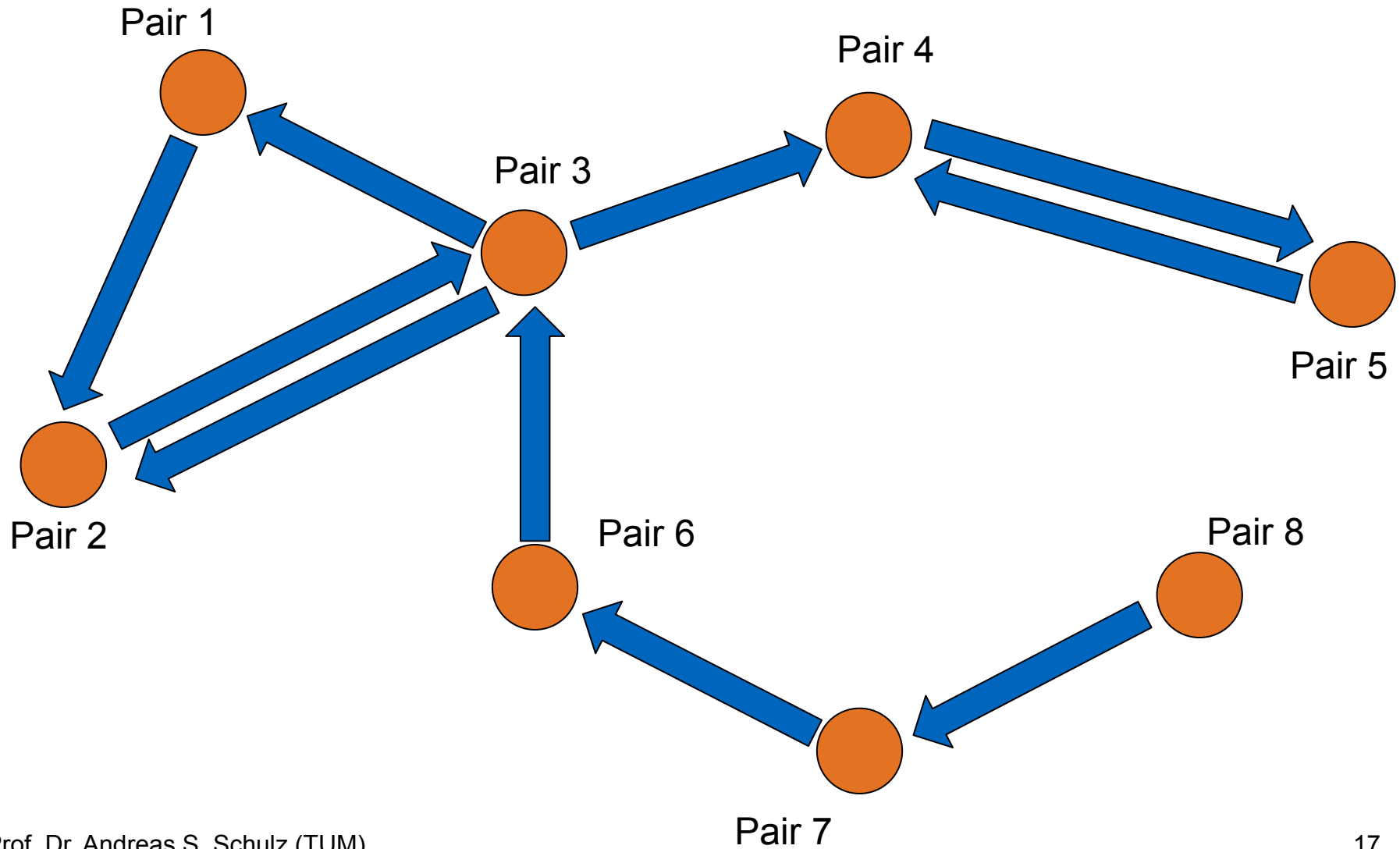
UNOS – initiated a national kidney exchange in the US  
Pilot in Oct 2010: 75+ hospitals registered

Methodist in San Antonio - single center program

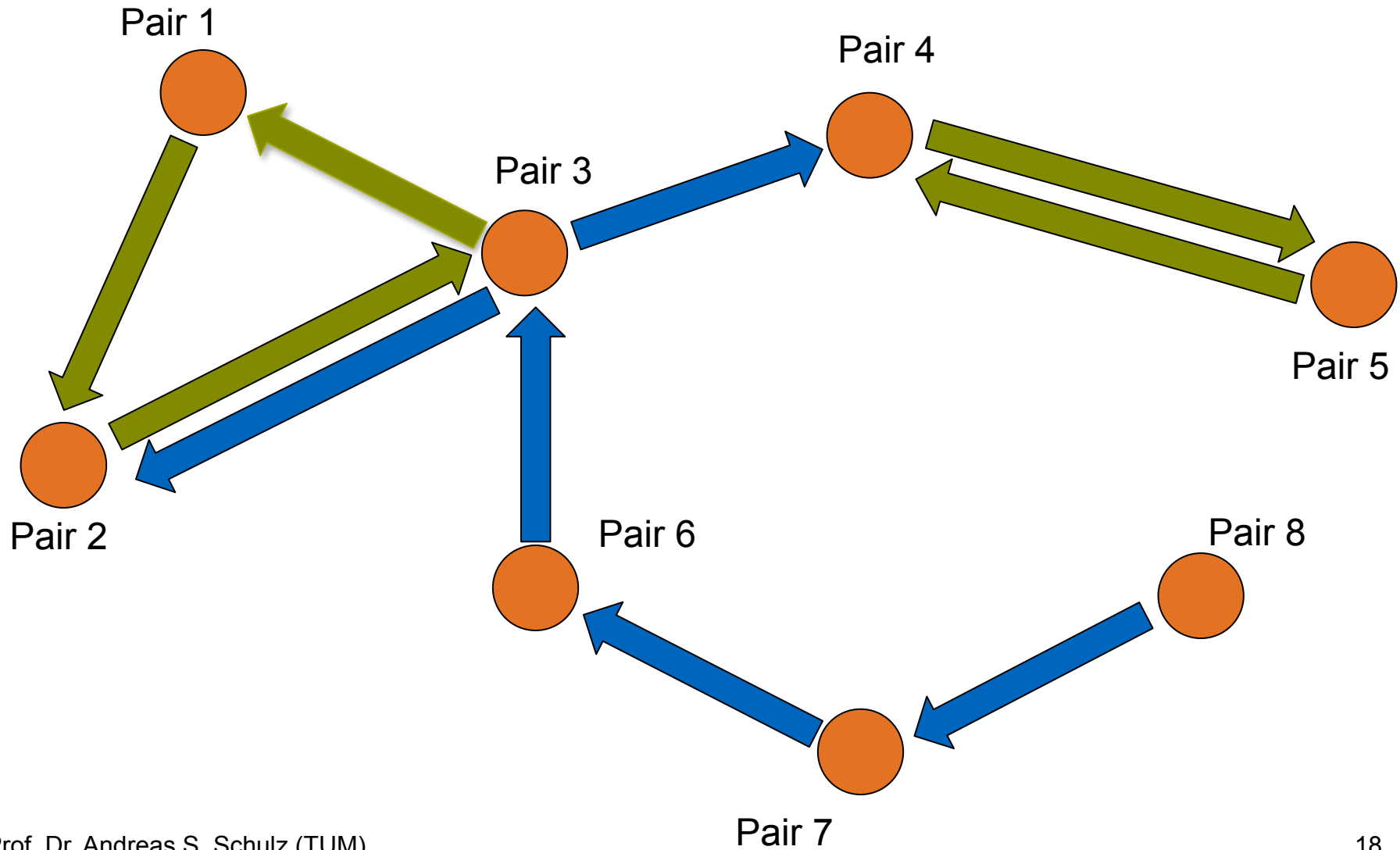
Hospitals enroll their incompatible pairs with their medical data.



# Compatibility graph



# Compatibility graph



# Challenge I: Maximize the number of matched pairs

Discrete optimization formulation...

$$\max \sum_{i,j} x_{i,j}$$

$$s.t. \quad x_{i,j} \leq a_{i,j}$$

$$\text{For all } i: \sum_j x_{i,j} = \sum_j x_{j,i}$$

$$\text{For all } i: \sum_j x_{i,j} \leq 1$$

$$x_{i,j} \text{ binary}$$

Challenge I: Maximize the number of matched pairs so that the largest cycle length is at most  $k$

Discrete optimization formulation...

Solve iteratively by eliminating cycles with length more than  $k$

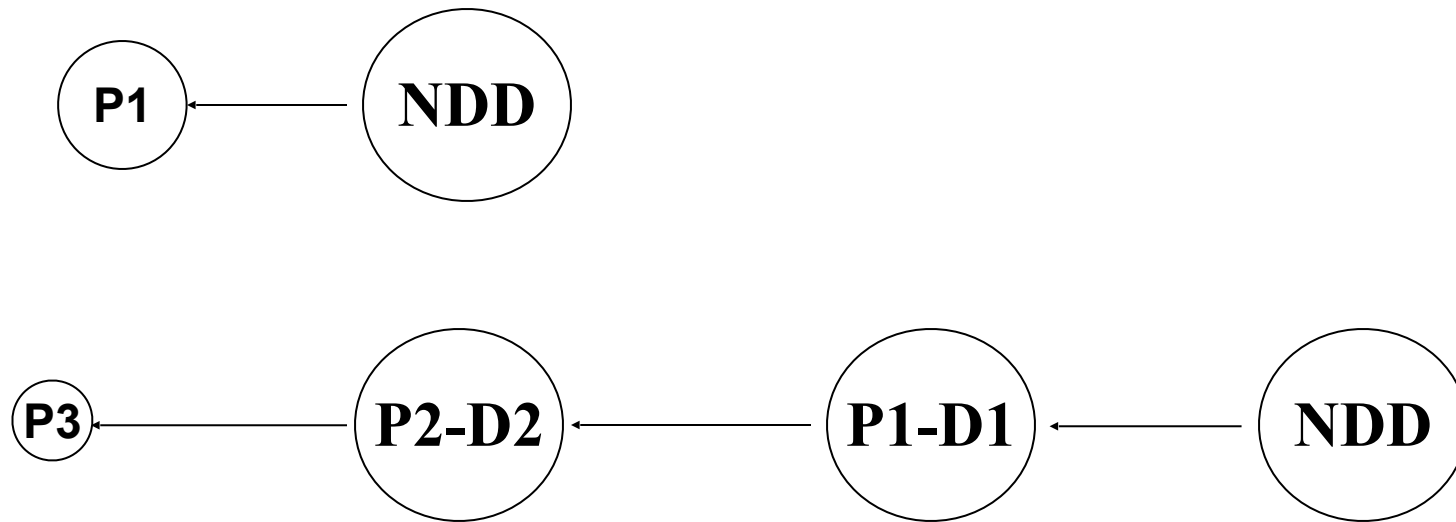
# Results

Max number of matched pairs

s.t. cycles are at most of size  $k$

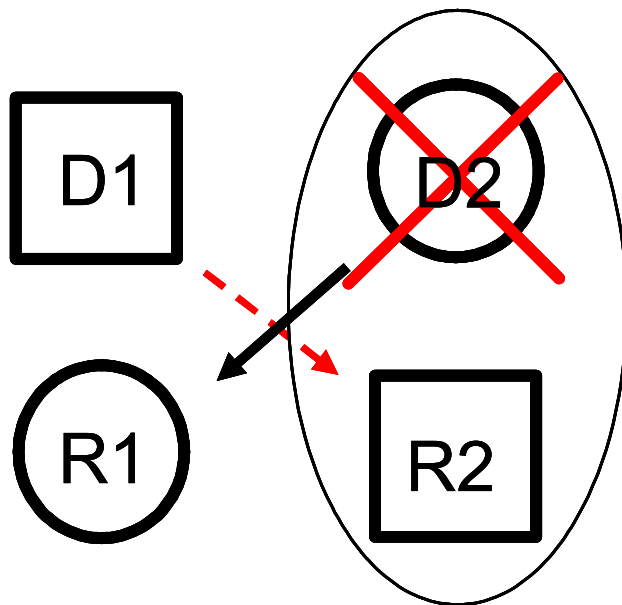
	Set size	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = \infty$
Historical 1st Set	361	118	169	189	191	193
1st Set (a)	131	6	9	11	13	16
1st Set (b)	251	8	12	16	16	16
2nd Set	74	22	42	45	47	51

# Non-Directed (altruistic) Donors

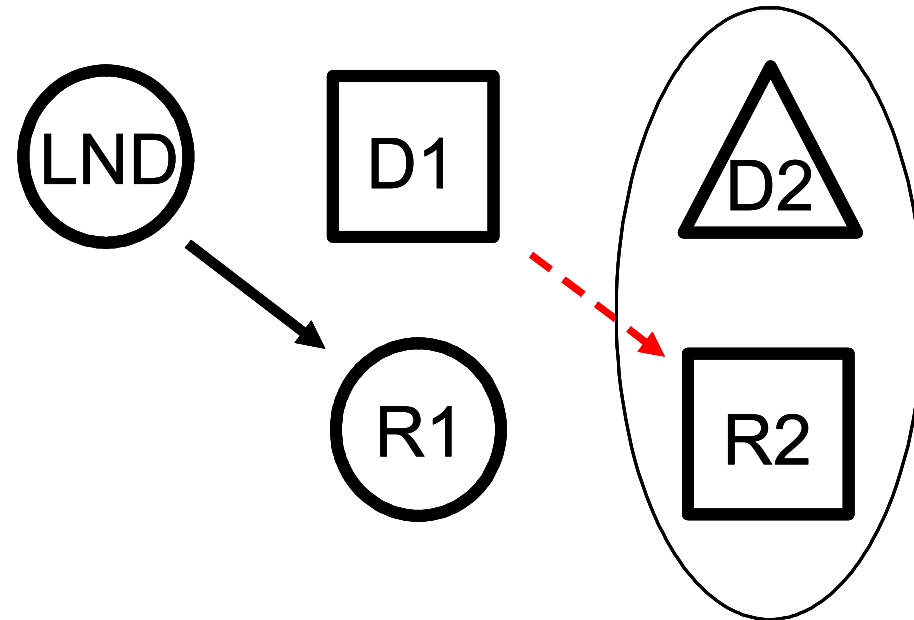


# Non-simultaneous extended altruistic donor chains

- Reduced risk from a broken link
- Since **NEAD** chains don't require simultaneity, they can be **longer**...

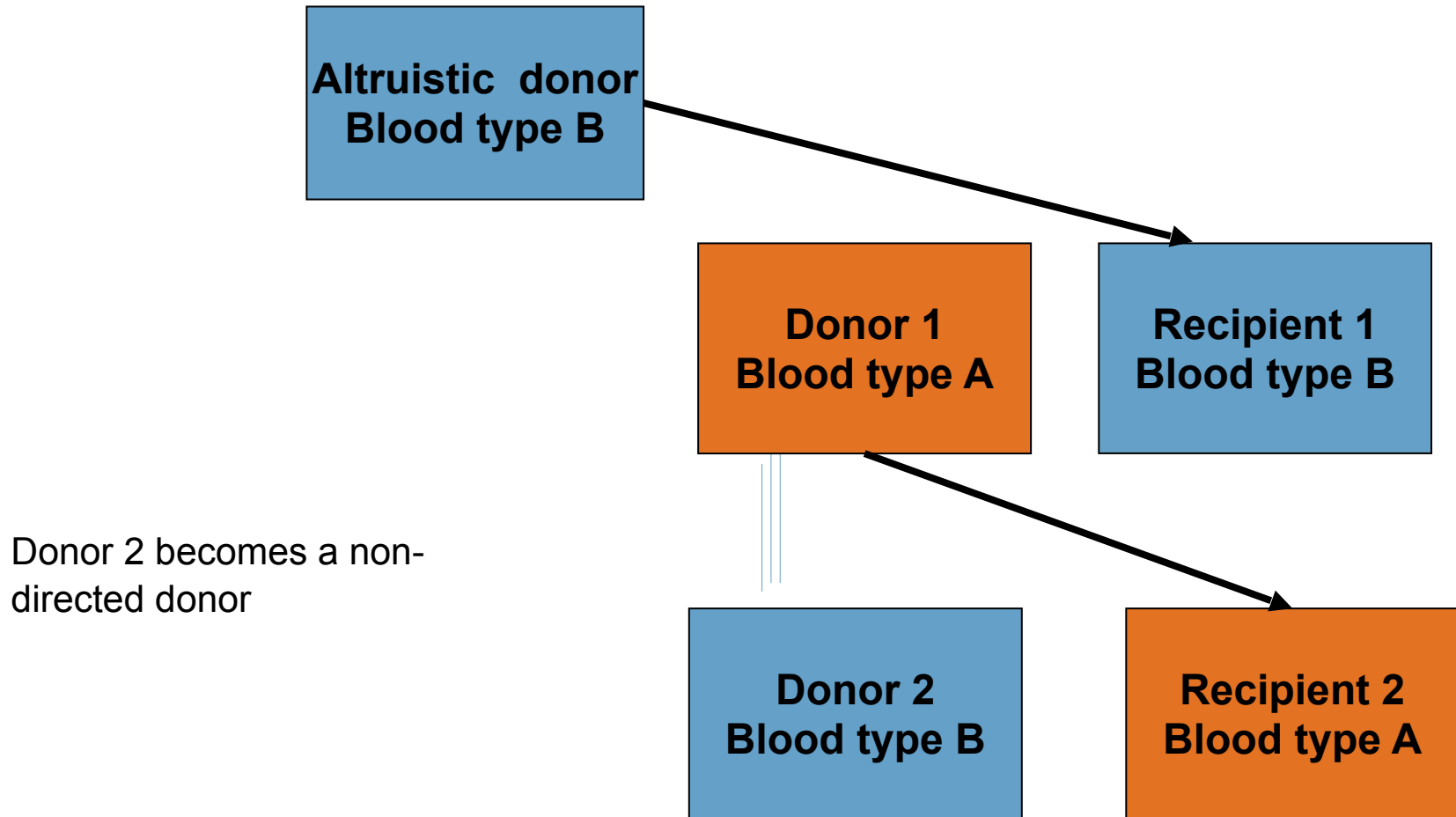


A. Conventional 2-way Matching



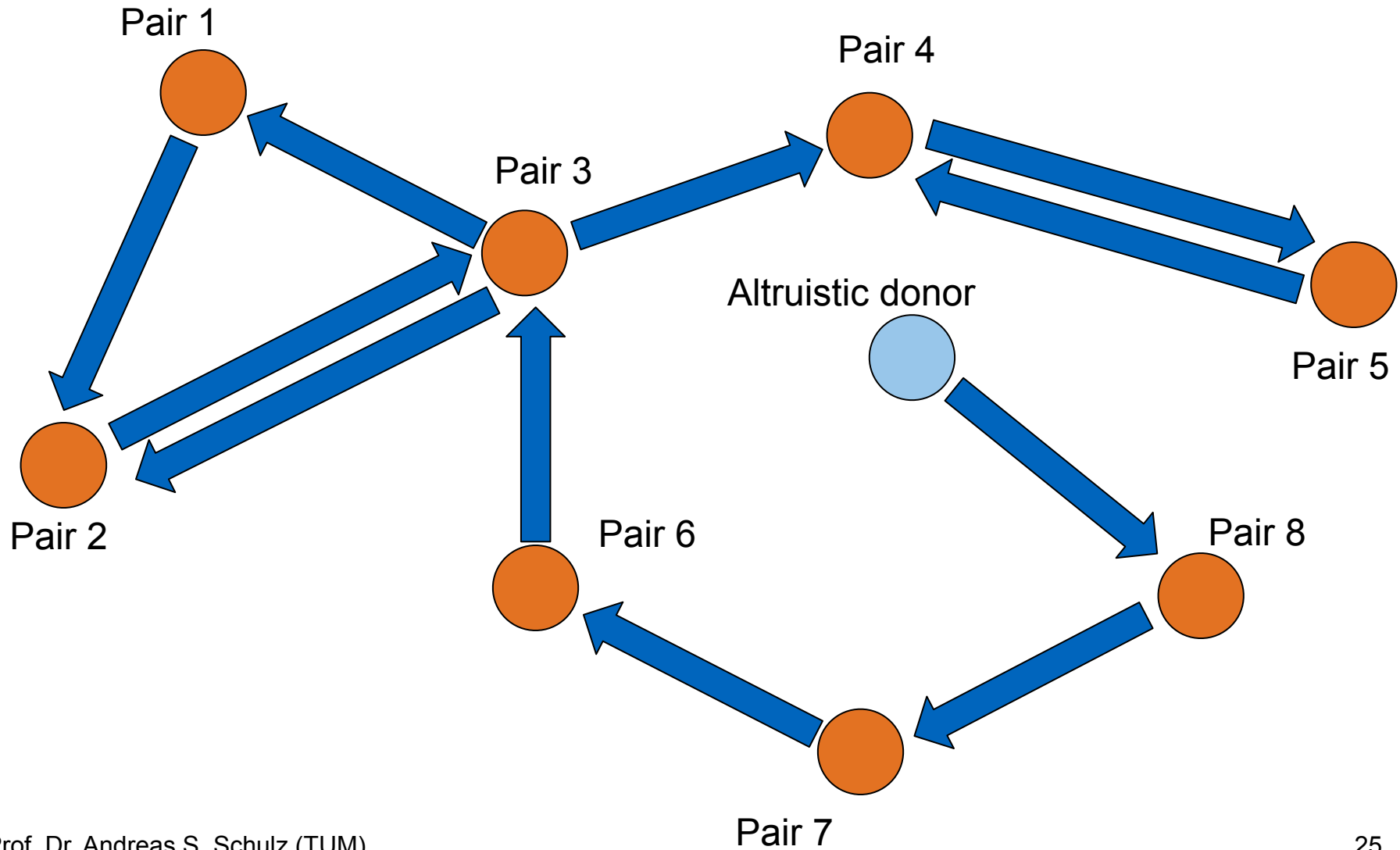
B. NEAD Chain Matching

# Non-Simultaneous Chain

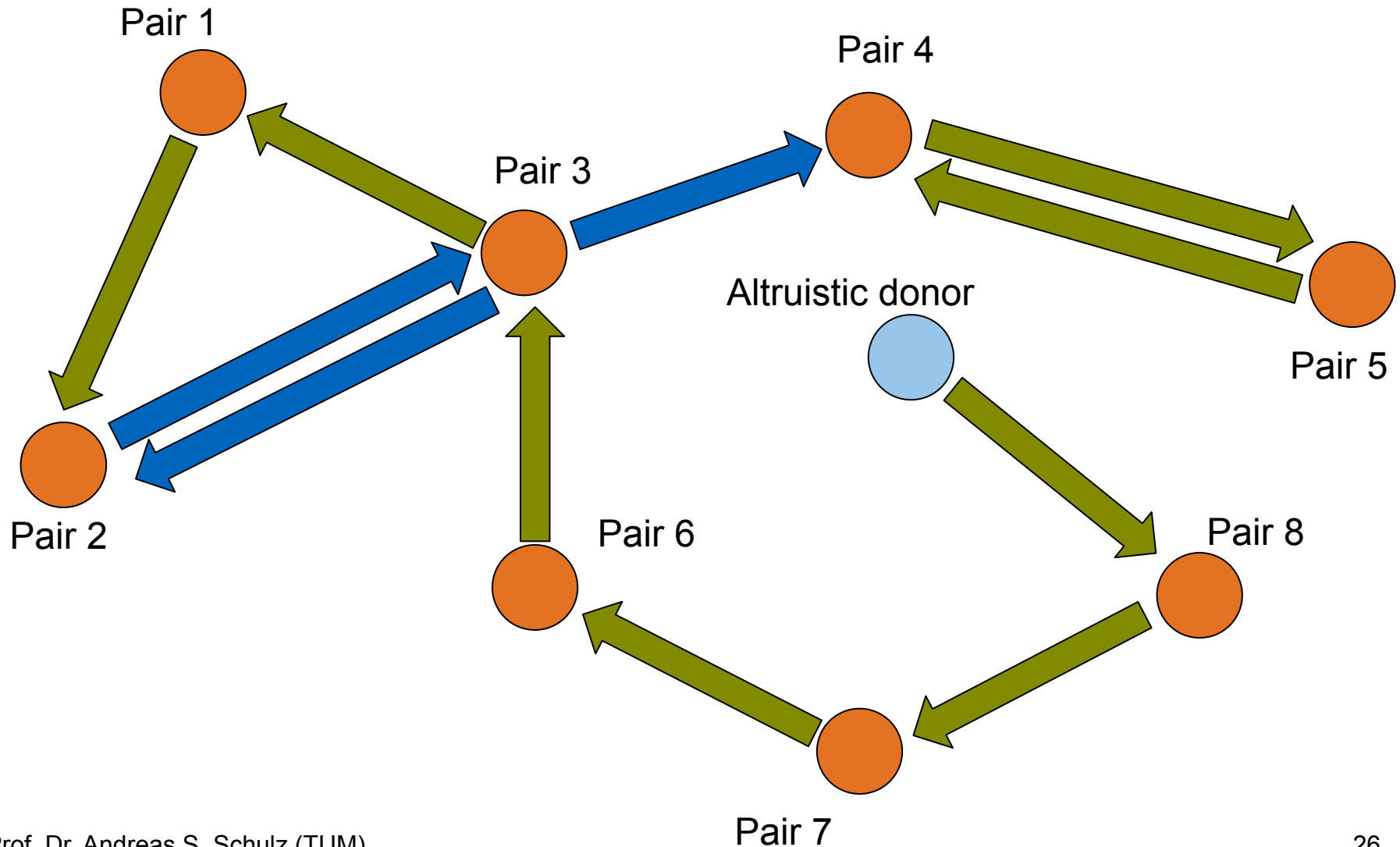




# Compatibility graph



# Compatibility graph



# The first long chain initiated by an altruistic donor

2009 HEROES AMONG US AWARDS



# 30 pairs chain (NKR)



# Challenge II: Maximize the number of matched pairs starting from an altruistic donor

Discrete optimization formulation. Longest path problem...

# Results

	No altruistic donors						One altruistic donor		
	Set size	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = \infty$	$(3, \infty)$	$(\infty, \infty)$	$(3, 5)$
Historical 1st Set	361	118	169	189	191	193	194.25	194.25	172
1st Set (a)	131	6	9	11	13	16	17.3	17.3	10.1
1st Set (b)	251	8	12	16	16	16	17.9	17.9	14.1
2nd Set	74	22	42	45	47	51	54.5	54.5	47.6



# Challenge III: What is (if any) the advantage of merging two programs into one bigger program?

Say two programs separately (NKR and APD) have one altruistic donor.

**NKR** can generate a chain of *23 pairs*

**APD** can generate a chain of *43 pairs*

Together they can generate two chains with lengths *54* and *45* (total *99* vs *66*).

# Maximize the number of transplants

A first discrete optimization formulation...

$$\begin{aligned}
 & \max \sum_{e \in E} w_e y_e \\
 & \text{s.t.} \quad \sum_{e \in \delta^-(v)} y_e = f_v^{in} & v \in V \\
 & \quad \sum_{e \in \delta^+(v)} y_e = f_v^{out} & v \in V \\
 & \quad f_v^0 \leq f_v^i \leq 1 & v \in P \\
 & \quad f_v^0 \leq 1 & v \in N \\
 & \quad y_e \in \{0,1\} & e \in E
 \end{aligned}$$

- $V$  is the union of the set of patients  $P$  and the set of altruistic donors  $N$ ,  $f$  are flow variables and  $y_e$  represents whether an arc is used, i.e., whether a certain transplantation is performed
- Can easily be solved by flow techniques
- Leads to cycles of unbounded length

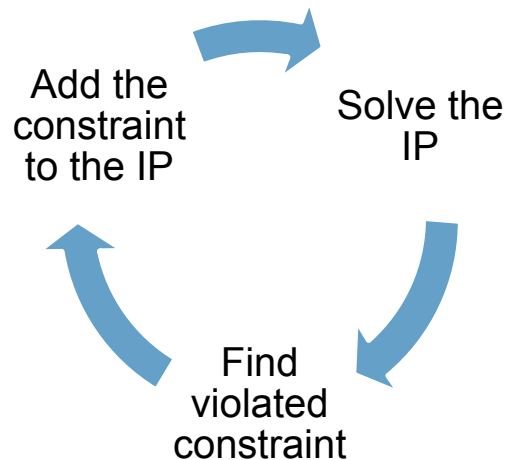


# Recursive Algorithm

$$\begin{aligned}
 & \max \sum_{e \in E} w_e y_e \\
 & \text{s.t.} \quad \sum_{e \in \delta^-(v)} y_e = f_v^i & v \in V \\
 & \quad \sum_{e \in \delta^+(v)} y_e = f_v^o & v \in V \\
 & \quad f_v^0 \leq f_v^i \leq 1 & v \in P \\
 & \quad f_v^0 \leq 1 & v \in N \\
 & \quad \sum_{e \in C} y_e \leq |C| - 1 & C \in \mathcal{C} \setminus \mathcal{C}_k \quad (1) \\
 & \quad y_e \in \{0,1\} & e \in E
 \end{aligned}$$

# Recursive Algorithm

- There are exponentially many constraints of type (1)
- “Constraint Generation”



$$\begin{aligned}
 & \max \sum_{e \in E} w_e y_e \\
 & \text{s.t.} \quad \sum_{e \in \delta^-(v)} y_e = f_v^i \quad v \in V \\
 & \quad \sum_{e \in \delta^+(v)} y_e = f_v^o \quad v \in V \\
 & \quad f_v^0 \leq f_v^i \leq 1 \quad v \in P \\
 & \quad f_v^0 \leq 1 \quad v \in N \\
 & \quad \sum_{e \in C} y_e \leq |C| - 1 \quad C \in \mathcal{C} \setminus \mathcal{C}_k \quad (1) \\
 & \quad y_e \in \{0,1\} \quad e \in E
 \end{aligned}$$

- The effectiveness depends on not having to generate too many constraints (1)
- Worst case: Exponentially many IPs have to be solved, each of which can take exponential time

# PC-TSP-Based Algorithm

$$\begin{aligned}
 & \max \sum_{e \in E} w_e y_e + \sum_{C \in \mathcal{C}_k} w_C z_C \\
 & \text{s.t.} \quad \sum_{e \in \delta^-(v)} y_e = f_v^i \quad v \in V \\
 & \quad \sum_{e \in \delta^+(v)} y_e = f_v^o \quad v \in V \\
 & \quad f_v^0 + \sum_{C \in \mathcal{C}_k(v)} z_C \leq f_v^i + \sum_{C \in \mathcal{C}_k(v)} z_C \leq 1 \quad v \in P \\
 & \quad f_v^0 \leq 1 \quad v \in N \\
 & \quad \sum_{e \in \delta^-(S)} y_e \geq f_v^i \quad S \subseteq P, v \in S \quad (2) \\
 & \quad y_e \in \{0,1\} \quad e \in E \\
 & \quad z_C \in \{0,1\} \quad C \in \mathcal{C}_k
 \end{aligned}$$

- Exponentially many constraints (2)
- Instead of solving the IP completely, want to add these constraints ‘on the fly’
  - Need to solve the *Separation Problem*
  - Can be solved by solving  $\mathcal{O}(|P|)$  flow problems

# Numerical Experiments

**Table S3. Performance of the recursive and TSP algorithms for “difficult” real-data KEP instances**

NDDs	Patient–donor pairs	Edges	Running time, s	
			Recursive	TSP
3	202	4,706	0.148	0.031
10	156	1,109	13.093	0.022
6	263	8,939	59.158	1.655
5	284	10,126	71.066	0.807
6	324	13,175	418.27	0.981
6	328	13,711	474.947	1.947
6	312	13,045	1,200*	0.157*
10	152	1,125	48.56	0.054
3	269	2,642	40.506	0.134
10	257	2,461	67.783	0.258
7	255	2,390	85.475	0.268
6	215	6,145	248.46	0.532
10	255	2,550	216.48	0.126
1	310	4,463	721.66	0.555
11	257	2,502	1,039.105	0.125
6	261	8,915	1,200	4.435
10	256	2,411	587.238	0.114
6	330	13,399	1,200	1.621
10	256	2,347	1,200	0.305
7	291	3,771	1,200*	0.221
8	275	3,158	1,200*	0.224
4	289	3,499	1,200*	0.2
3	199	2,581	1,200*	0.041
7	198	4,882	1,200*	8.204
2	389	8,346	1,200*	0.096

Timeouts (optimal solution not found) are indicated by an asterisk.


- The PC-TSP-based algorithm performs much better than the recursive algorithm

# Chains are important for hard to match pairs

As kidney exchange became common and transplant centers gained experience they began withholding their easy match pairs and transplanting them internally



The flow of new patients to kidney exchange networks contains many who are hard to match



So chains become important: many pairs with few compatible kidneys can only be reached through chains

# Highly Sensitized Patients

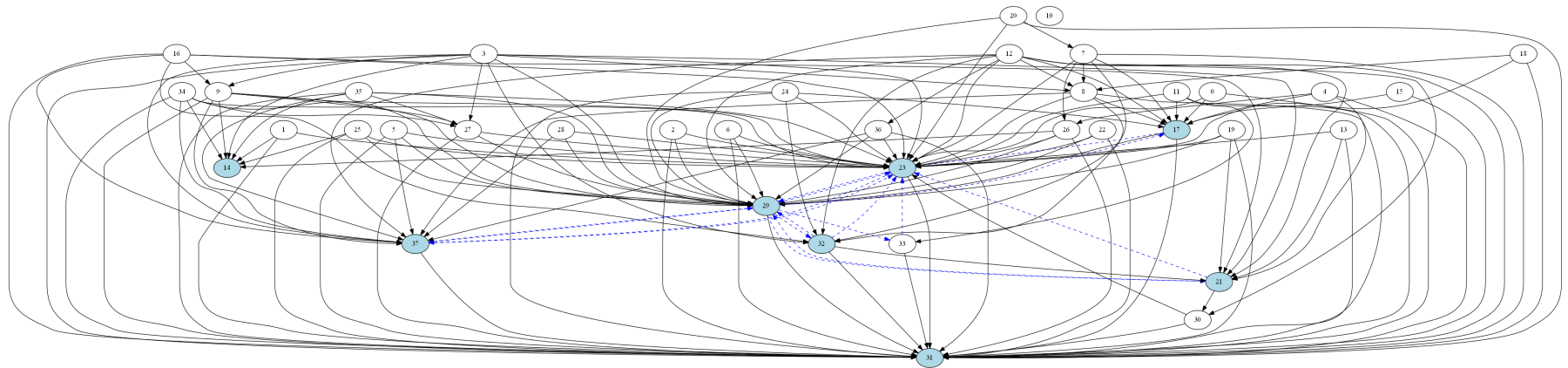


More than 14% are immunologically compatible with fewer than 1% of kidneys. They are looking for a needle in a haystack



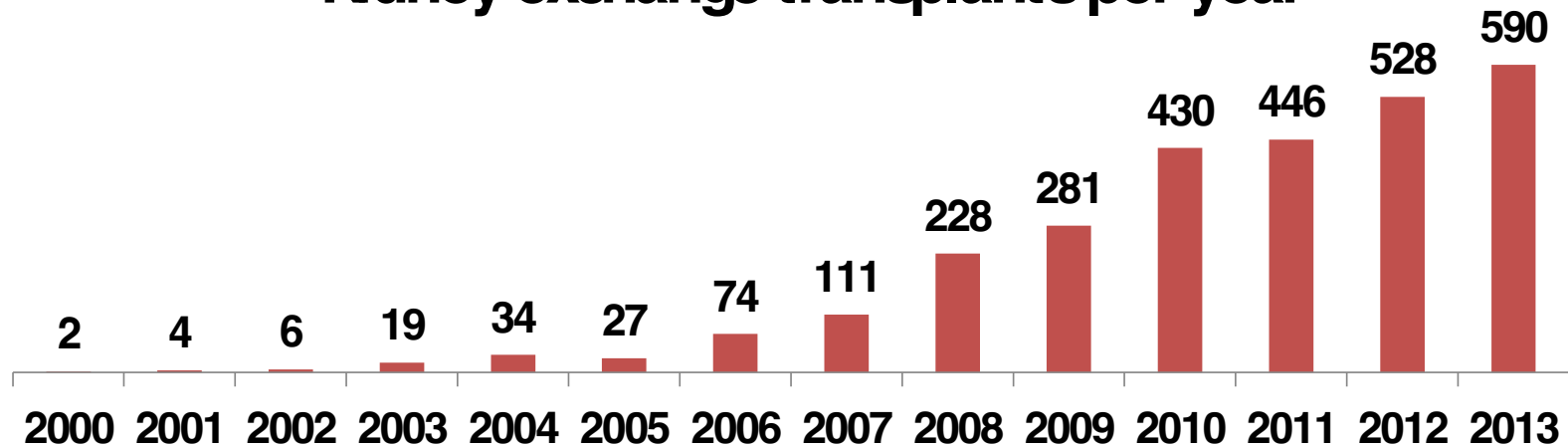
Very highly sensitized patients had a median waiting time of >13 years for a deceased donor kidney

# Example Graph from Data



# Growth of kidney exchange in the US

## Kidney exchange transplants per year



**..5-fold increase since 2007, majority now in chains**



# Wrap-up

- Discrete optimization provides a useful quantified way of matching better and more incompatible pairs
- Discrete optimization provides insights into the value added by altruistic donors and advantages of merging different exchange programs into one program

## Non-directed donors: cycles plus chains

