A 3/2-approximation algorithm for the student-project allocation problem with ties

Frances Cooper
Joint work with: Dr David Manlove
Outline
Outline

• Matching problems
Outline

• Matching problems

• Maximum sized stable matching
Outline

• Matching problems

• Maximum sized stable matching
  • Integer programming
Outline

• Matching problems

• Maximum sized stable matching
  • Integer programming
  • Approximation algorithm
Outline

• Matching problems

• Maximum sized stable matching
  • Integer programming
  • Approximation algorithm

• Future work
Matching Problems
Matching Problems

• Assign one set of entities to another set of entities
Matching Problems

- Assign one set of entities to another set of entities
- Based on preferences and capacities
Student-project allocation problem (SPA-ST)
Student-project allocation problem (SPA-ST)
Student-project allocation problem (SPA-ST)
Student-project allocation problem (SPA-ST)
Student-project allocation problem (SPA-ST)

Students

- s1
- s2
- s3
- s4

Projects

- p1
- p2
- p3

Lecturers

- l1
- l2
Student-project allocation problem (SPA-ST)

- **Students**
  - s1
  - s2
  - s3
  - s4

- **Projects**
  - p1
  - p2
  - p3

- **Lecturers**
  - l1
  - l2
Student-project allocation problem (SPA-ST)

Students

- s1
- s2
- s3
- s4

Projects

- p1: 1 space
- p2: 2 spaces
- p3: 1 space

Lecturers

- l1: 2 spaces
- l2: 2 spaces
Student-project allocation problem (SPA-ST)
Student-project allocation problem (SPA-ST)
Student-project allocation problem (SPA-ST)
Stable matching
Stable matching

- weak stability
- strong stability
- super stability
A stable matching is a matching with no blocking pairs.
A stable matching is a matching with no blocking pairs.

- **Weak stability**: No Blocking pairs: both agents are better off.
- **Strong stability**: No Blocking Pairs: one agent is better off, the other is no worse off.
- **Super stability**: No Blocking Pairs: neither agent is worse off.
A stable matching is a matching with no blocking pairs.

- **weak stability**: No Blocking pairs: both agents are better off
- **strong stability**: No Blocking Pairs: one agent is better off, the other is no worse off
- **super stability**: No Blocking Pairs: neither agent is worse off
weak stability
weak stability

Blocking pair: both agents are better off
weak stability

Blocking pair: both agents are better off

project and lecturer undersubscribed
weak stability

Blocking pair: both agents are better off

project and lecturer undersubscribed

project undersubscribed, lecturer full
weak stability

Blocking pair: both agents are better off

Project and lecturer undersubscribed

Project undersubscribed, lecturer full

Project full, (lecturer full or undersubscribed)
Maximum stable matchings
Maximum stable matchings

- A **stable matching** is a matching with no blocking pairs
Maximum stable matchings

- A **stable matching** is a matching with no blocking pairs
- No ties in preference lists - find a stable matching in polynomial time - all same size
Maximum stable matchings

- **A stable matching** is a matching with no blocking pairs

- No ties in preference lists - find a stable matching in polynomial time - all same size

- **Ties in preference lists** - find a stable matching in polynomial time - but stable matchings are different sizes
Maximum stable matchings

• A **stable matching** is a matching with no blocking pairs

• No ties in preference lists - find a stable matching in polynomial time - all same size

• **Ties in preference lists** - find a stable matching in polynomial time - but stable matchings are **different sizes**

• Finding a maximum sized stable matching is **NP-hard**.
Finding a maximum sized stable matching
Finding a maximum sized stable matching

Two techniques:
Finding a maximum sized stable matching

Two techniques:

1. Approximation algorithm
Finding a maximum sized stable matching

Two techniques:

1. Approximation algorithm
2. Integer Programming
Approximation Algorithm
Previous work
Previous work

- Hospitals/Residents with Ties (HRT) - special case of SPA-ST, each lecturer offers one project and the capacity of each lecturer equals the capacity of their offered project
Previous work

• Hospitals/Residents with Ties (HRT) - special case of SPA-ST, each lecturer offers one project and the capacity of each lecturer equals the capacity of their offered project

• A 3/2-approximation algorithm exists for HRT

Linear Time Local Approximation Algorithm for Maximum Stable Marriage; Algorithms; 2013; Kiraly
Previous work

- Hospitals/Residents with Ties (HRT) - special case of SPA-ST, each lecturer offers one project and the capacity of each lecturer equals the capacity of their offered project

- A 3/2-approximation algorithm exists for HRT

- Can I just convert my problem and use this algorithm?

Linear Time Local Approximation Algorithm for Maximum Stable Marriage; Algorithms; 2013; Kiraly
Previous work

- Hospitals/Residents with Ties (HRT) - special case of SPA-ST, each lecturer offers one project and the capacity of each lecturer equals the capacity of their offered project
- A 3/2-approximation algorithm exists for HRT
- Can I just convert my problem and use this algorithm?
- Not using a conversion process we tried.

Linear Time Local Approximation Algorithm for Maximum Stable Marriage; Algorithms; 2013; Kiraly
3/2-approximation algorithm
3/2-approximation algorithm

- Created a new 3/2 approximation algorithm for SPA-ST, based on Kiraly’s HRT algorithm.
3/2-approximation algorithm

• Created a new 3/2 approximation algorithm for SPA-ST, based on Kiraly’s HRT algorithm.

• Moving from HRT to SPA-ST
3/2-approximation algorithm

• Created a new 3/2 approximation algorithm for SPA-ST, based on Kiraly’s HRT algorithm.
  • Moving from HRT to SPA-ST
    • Lecturers added a lot of complications
3/2-approximation algorithm

- Created a new 3/2 approximation algorithm for SPA-ST, based on Kiraly’s HRT algorithm.
  - Moving from HRT to SPA-ST
    - Lecturers added a lot of complications
    - Definition of a blocking pair is more complicated
Approximation algorithm
high-level look
Approximation algorithm
high-level look

Students (who are not already assigned) apply in turn to their favourite project on their preference list. Assume student $s$ applies to project $p$. 
Approximation algorithm  
high-level look

Students (who are not already assigned) apply in turn to their favourite project on their preference list. Assume student $s$ applies to project $p$.

- if $p$ and $l$ (the lecturer of $p$) are undersubscribed then we add $(s,p)$ to our matching
Approximation algorithm
high-level look

Students (who are not already assigned) apply in turn to their favourite project on their preference list. Assume student $s$ applies to project $p$.

- if $p$ and $l$ (the lecturer of $p$) are undersubscribed then we add $(s,p)$ to our matching

- if either $p$ or $l$ are full then we need to check whether $(s,p)$ should replace an *existing* pair in the matching
Approximation algorithm
high-level look

Students (who are not already assigned) apply in turn to their favourite project on their preference list. Assume student $s$ applies to project $p$.

- if $p$ and $l$ (the lecturer of $p$) are undersubscribed then we add $(s,p)$ to our matching

- if either $p$ or $l$ are full then we need to check whether $(s,p)$ should replace an existing pair in the matching

- if there is no chance for $s$ to assign to $p$ then $s$ will remove $p$ from their preference list (and will now apply to their next favourite)
Approximation algorithm high-level look

Students (who are not already assigned) apply in turn to their favourite project on their preference list. Assume student \( s \) applies to project \( p \).

- if \( p \) and \( l \) (the lecturer of \( p \)) are undersubscribed then we add \((s,p)\) to our matching

- if either \( p \) or \( l \) are full then we need to check whether \((s,p)\) should replace an existing pair in the matching

- if there is no chance for \( s \) to assign to \( p \) then \( s \) will remove \( p \) from their preference list (and will now apply to their next favourite)

- Students iterate twice through their preference list
Proofs
Proofs

Three proofs required:
Proofs

Three proofs required:

• the algorithm runs in linear time
Proofs

Three proofs required:

• the algorithm runs in linear time

• the resultant matching is stable
Proofs

Three proofs required:

• the algorithm runs in linear time

• the resultant matching is stable

• the matching is at least $2/3$ the size of optimal
Performance guarantee - creating $G'$
Performance guarantee - creating $G'$
Performance guarantee - creating $G'$
Performance guarantee - creating $G'$

$G$

$G'$

$M_{opt}$

$M$

students

1

2

3

4
Performance guarantee - creating $G'$

$G$

$G'$

$M_{opt}$

$M$
Performance guarantee - creating $G'$

$G$

$s1$ $p1$ $l1$
$s2$ $p2$ $l1$
$s3$ $p3$ $l2$
$s4$ $p1$ $l2$

$G'$

students
1
2
3
4

lecturer clones
1
2
3
4

$M_{opt}$

$M$
Performance guarantee - creating $G'$
Performance guarantee - creating $G'$
Performance guarantee - creating $G'$

$G$

$s1$  $p1$  $l1$

$s2$  $p2$  $l2$

$s3$  $p3$

$s4$

$2$ spaces  $2$ spaces

$1$ space  $2$ spaces

$G'$

$G$

$1$

$2$

$3$

$4$

students

lecturer clones

$M_{opt}$

$M$
Performance guarantee - creating $G'$
Performance guarantee - creating $G'$

$G$

- $s1$ connected to $p1$, $l1$ with 2 spaces
- $s2$ connected to $p1$, $l1$ with 2 spaces
- $s3$ connected to $p2$, $l2$ with 1 space
- $s4$ connected to $p3$, $l2$ with 1 space

$G'$

- $s1$ connected to $p1$, $l1$ with 2 spaces
- $s2$ connected to $p1$, $l1$ with 2 spaces
- $s3$ connected to $p2$, $l2$ with 1 space
- $s4$ connected to $p3$, $l2$ with 1 space

---

$M_{opt}$

$M'$

Frances Cooper
Structures in $G'$
Structures in $G'$

• odd length alternating path with end edges in $M'_{opt}$
  (number of edges is 3)
Structures in $G'$

- odd length alternating path with end edges in $M'_{opt}$
  (number of edges is 3)

```
students

1 - 1

2 - 2

lecturer clones
```
Structures in $G'$

- odd length alternating path with end edges in $M_{opt}'$ (number of edges is 3)

- odd length alternating path with end edges in $M_{opt}'$ (number of edges is 1)
Structures in $G'$

- odd length alternating path with end edges in $M'_{opt}$ (number of edges is 3)

- odd length alternating path with end edges in $M'_{opt}$ (number of edges is 1)
Structures in $G$
Structures in $G$
Structures in \( G \)
Structures in $G'$
Structures in $G'$

Diagram of students and lecturers with connections labeled as $s1$, $p1$, $l1$, $s2$, $p2$, $l2$, $s1$, $p1$, $l1$, $s2$, $p2$, $l2$, $s1$, $p2$, $l1$, and $s2$, $p3$, $l2$. The diagram illustrates the relationships between students, lecturers, and clones in the context of the graph $G'$. The nodes are connected with lines indicating the relationships and interactions.
Structures in $G'$

- Students: $s_1, s_2$
- Lecturers: $l_1, l_2$
- Clones: $p_1, p_2, p_3$

Diagram shows the relationships between the students, lecturers, and clones in graph $G'$.
Structures in $G'$
Structures in $G$
Integer Program
Integer Programming
Integer Programming
Integer Programming

• gives an optimal solution
Integer Programming

• gives an optimal solution

• novel work: stability constraints
Integer Programming

• gives an optimal solution

• novel work: stability constraints

• helped in correctness checking
Integer Programming

- gives an optimal solution
- novel work: stability constraints
- helped in correctness checking
- gives motivation for using approximation algorithm
Experimental results
Experimental Results
Experimental Results

- Java (and Gurobi), 100s of thousands of instances with varying parameters. Ran on approximation algorithm and integer program.
Experimental Results

- Java (and Gurobi), 100s of thousands of instances with varying parameters. Ran on approximation algorithm and integer program.

- Does the approximation algorithm stick to 2/3 the size of optimal? Or do we get close to maximum?
Experimental Results

• Java (and Gurobi), 100s of thousands of instances with varying parameters. Ran on approximation algorithm and integer program.

• Does the approximation algorithm stick to 2/3 the size of optimal? Or do we get close to maximum?

<table>
<thead>
<tr>
<th>Case</th>
<th>minimum A/Max</th>
<th>average size A/Max</th>
<th>Min/Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIES1</td>
<td>1.0000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>TIES2</td>
<td>0.9792</td>
<td>0.997</td>
<td>0.987</td>
</tr>
<tr>
<td>TIES3</td>
<td>0.9722</td>
<td>0.993</td>
<td>0.972</td>
</tr>
<tr>
<td>TIES4</td>
<td>0.9655</td>
<td>0.990</td>
<td>0.958</td>
</tr>
<tr>
<td>TIES5</td>
<td>0.9626</td>
<td>0.986</td>
<td>0.942</td>
</tr>
<tr>
<td>TIES6</td>
<td>0.9558</td>
<td>0.984</td>
<td>0.927</td>
</tr>
<tr>
<td>TIES7</td>
<td>0.9486</td>
<td>0.982</td>
<td>0.911</td>
</tr>
<tr>
<td>TIES8</td>
<td>0.9527</td>
<td>0.980</td>
<td>0.896</td>
</tr>
<tr>
<td>TIES9</td>
<td>0.9467</td>
<td>0.980</td>
<td>0.880</td>
</tr>
<tr>
<td>TIES10</td>
<td>0.9529</td>
<td>0.982</td>
<td>0.866</td>
</tr>
<tr>
<td>TIES11</td>
<td>0.9467</td>
<td>0.984</td>
<td>0.851</td>
</tr>
</tbody>
</table>

• TIES - 10,000 instances per set, 300 students, 250 projects (capacity 420), 120 lecturers (capacity 360), pref lists length 3 to 5.
Experimental Results

- Java (and Gurobi), 100s of thousands of instances with varying parameters. Ran on approximation algorithm and integer program.

- Does the approximation algorithm stick to 2/3 the size of optimal? Or do we get close to maximum?

<table>
<thead>
<tr>
<th>Case</th>
<th>minimum A/Max</th>
<th>average A/Max</th>
<th>Min/Max A/Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIES1</td>
<td>1.0000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>TIES2</td>
<td>0.9792</td>
<td>0.997</td>
<td>0.987</td>
</tr>
<tr>
<td>TIES3</td>
<td>0.9722</td>
<td>0.993</td>
<td>0.972</td>
</tr>
<tr>
<td>TIES4</td>
<td>0.9655</td>
<td>0.990</td>
<td>0.958</td>
</tr>
<tr>
<td>TIES5</td>
<td>0.9626</td>
<td>0.986</td>
<td>0.942</td>
</tr>
<tr>
<td>TIES6</td>
<td>0.9558</td>
<td>0.984</td>
<td>0.927</td>
</tr>
<tr>
<td>TIES7</td>
<td>0.9486</td>
<td>0.982</td>
<td>0.911</td>
</tr>
<tr>
<td>TIES8</td>
<td>0.9527</td>
<td>0.980</td>
<td>0.896</td>
</tr>
<tr>
<td>TIES9</td>
<td>0.9467</td>
<td>0.980</td>
<td>0.880</td>
</tr>
<tr>
<td>TIES10</td>
<td>0.9529</td>
<td>0.982</td>
<td>0.866</td>
</tr>
<tr>
<td>TIES11</td>
<td>0.9467</td>
<td>0.984</td>
<td>0.851</td>
</tr>
</tbody>
</table>

- TIES - 10,000 instances per set, 300 students, 250 projects (capacity 420), 120 lecturers (capacity 360), pref lists length 3 to 5.

- increasing prob of student and lecturer ties from 0 to 0.5 in 0.05 steps
Experimental Results

• Java (and Gurobi), 100s of thousands of instances with varying parameters. Ran on approximation algorithm and integer program.

• Does the approximation algorithm stick to 2/3 the size of optimal? Or do we get close to maximum?

<table>
<thead>
<tr>
<th>Case</th>
<th>minimum A/Max</th>
<th>average size A/Max</th>
<th>Min/Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIES1</td>
<td>1.0000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>TIES2</td>
<td>0.9792</td>
<td>0.997</td>
<td>0.987</td>
</tr>
<tr>
<td>TIES3</td>
<td>0.9722</td>
<td>0.993</td>
<td>0.972</td>
</tr>
<tr>
<td>TIES4</td>
<td>0.9655</td>
<td>0.990</td>
<td>0.958</td>
</tr>
<tr>
<td>TIES5</td>
<td>0.9626</td>
<td>0.986</td>
<td>0.942</td>
</tr>
<tr>
<td>TIES6</td>
<td>0.9558</td>
<td>0.984</td>
<td>0.927</td>
</tr>
<tr>
<td>TIES7</td>
<td>0.9486</td>
<td>0.982</td>
<td>0.911</td>
</tr>
<tr>
<td>TIES8</td>
<td>0.9527</td>
<td>0.980</td>
<td>0.896</td>
</tr>
<tr>
<td>TIES9</td>
<td>0.9467</td>
<td>0.980</td>
<td>0.880</td>
</tr>
<tr>
<td>TIES10</td>
<td>0.9529</td>
<td>0.982</td>
<td>0.866</td>
</tr>
<tr>
<td>TIES11</td>
<td>0.9467</td>
<td>0.984</td>
<td>0.851</td>
</tr>
</tbody>
</table>

• TIES - 10,000 instances per set, 300 students, 250 projects (capacity 420), 120 lecturers (capacity 360), pref lists length 3 to 5.

• Increasing prob of student and lecturer ties from 0 to 0.5 in 0.05 steps

• Average approx solution closer to optimal than minimum in all cases
Experimental Results
Experimental Results

Scalability

<table>
<thead>
<tr>
<th>Case</th>
<th>A</th>
<th>Max</th>
<th>A</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCALS1</td>
<td>10</td>
<td>10</td>
<td>1393.8</td>
<td>227764.3</td>
</tr>
<tr>
<td>SCALS2</td>
<td>10</td>
<td>9</td>
<td>5356.7</td>
<td>1096045.6</td>
</tr>
<tr>
<td>SCALS3</td>
<td>10</td>
<td>0</td>
<td>13095.3</td>
<td>N/A</td>
</tr>
<tr>
<td>SCALS4</td>
<td>10</td>
<td>0</td>
<td>18883.5</td>
<td>N/A</td>
</tr>
<tr>
<td>SCALS5</td>
<td>10</td>
<td>0</td>
<td>20993.0</td>
<td>N/A</td>
</tr>
<tr>
<td>SCALP1</td>
<td>10</td>
<td>9</td>
<td>193.3</td>
<td>94242.9</td>
</tr>
<tr>
<td>SCALP2</td>
<td>10</td>
<td>10</td>
<td>189.4</td>
<td>631225.2</td>
</tr>
<tr>
<td>SCALP3</td>
<td>10</td>
<td>3</td>
<td>196.6</td>
<td>882251.0</td>
</tr>
<tr>
<td>SCALP4</td>
<td>10</td>
<td>1</td>
<td>248.5</td>
<td>1594201.0</td>
</tr>
<tr>
<td>SCALP5</td>
<td>10</td>
<td>0</td>
<td>283.7</td>
<td>N/A</td>
</tr>
<tr>
<td>SCALP6</td>
<td>10</td>
<td>0</td>
<td>288.4</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Experimental Results

Scalability

- SCALS - 10,000 students up to 50,000 students. Pref lists 3 to 5 and ties 0.2

<table>
<thead>
<tr>
<th>Case</th>
<th>A</th>
<th>Max</th>
<th>A</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCALS1</td>
<td>10</td>
<td>10</td>
<td>1393.8</td>
<td>227764.3</td>
</tr>
<tr>
<td>SCALS2</td>
<td>10</td>
<td>9</td>
<td>5356.7</td>
<td>1096045.6</td>
</tr>
<tr>
<td>SCALS3</td>
<td>10</td>
<td>0</td>
<td>13095.3</td>
<td>N/A</td>
</tr>
<tr>
<td>SCALS4</td>
<td>10</td>
<td>0</td>
<td>18883.5</td>
<td>N/A</td>
</tr>
<tr>
<td>SCALS5</td>
<td>10</td>
<td>0</td>
<td>20993.0</td>
<td>N/A</td>
</tr>
<tr>
<td>SCALP1</td>
<td>10</td>
<td>9</td>
<td>193.3</td>
<td>94242.9</td>
</tr>
<tr>
<td>SCALP2</td>
<td>10</td>
<td>10</td>
<td>189.4</td>
<td>631225.2</td>
</tr>
<tr>
<td>SCALP3</td>
<td>10</td>
<td>3</td>
<td>196.6</td>
<td>882251.0</td>
</tr>
<tr>
<td>SCALP4</td>
<td>10</td>
<td>1</td>
<td>248.5</td>
<td>1594201.0</td>
</tr>
<tr>
<td>SCALP5</td>
<td>10</td>
<td>0</td>
<td>283.7</td>
<td>N/A</td>
</tr>
<tr>
<td>SCALP6</td>
<td>10</td>
<td>0</td>
<td>288.4</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Experimental Results

Scalability

- SCALS - 10,000 students up to 50,000 students. Pref lists 3 to 5 and ties 0.2

- SCALP - 500 students, ties 0.4, Pref lists increased from 25 to 150 in steps of 25.

<table>
<thead>
<tr>
<th>Case</th>
<th>A</th>
<th>Max</th>
<th>A</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCALS1</td>
<td>10</td>
<td>10</td>
<td>1393.8</td>
<td>227764.3</td>
</tr>
<tr>
<td>SCALS2</td>
<td>10</td>
<td>9</td>
<td>5356.7</td>
<td>1096045.6</td>
</tr>
<tr>
<td>SCALS3</td>
<td>10</td>
<td>0</td>
<td>13095.3</td>
<td>N/A</td>
</tr>
<tr>
<td>SCALS4</td>
<td>10</td>
<td>0</td>
<td>18883.5</td>
<td>N/A</td>
</tr>
<tr>
<td>SCALS5</td>
<td>10</td>
<td>0</td>
<td>20993.0</td>
<td>N/A</td>
</tr>
<tr>
<td>SCALP1</td>
<td>10</td>
<td>9</td>
<td>193.3</td>
<td>94242.9</td>
</tr>
<tr>
<td>SCALP2</td>
<td>10</td>
<td>10</td>
<td>189.4</td>
<td>631225.2</td>
</tr>
<tr>
<td>SCALP3</td>
<td>10</td>
<td>3</td>
<td>196.6</td>
<td>882251.0</td>
</tr>
<tr>
<td>SCALP4</td>
<td>10</td>
<td>1</td>
<td>248.5</td>
<td>1594201.0</td>
</tr>
<tr>
<td>SCALP5</td>
<td>10</td>
<td>0</td>
<td>283.7</td>
<td>N/A</td>
</tr>
<tr>
<td>SCALP6</td>
<td>10</td>
<td>0</td>
<td>288.4</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Experimental Results

Scalability

- SCALS - 10,000 students up to 50,000 students. Pref lists 3 to 5 and ties 0.2

- SCALP - 500 students, ties 0.4, Pref lists increased from 25 to 150 in steps of 25.

- much faster than using the integer program

<table>
<thead>
<tr>
<th>Case</th>
<th>A</th>
<th>Max</th>
<th>A</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCALS1</td>
<td>10</td>
<td>10</td>
<td>1393.8</td>
<td>227764.3</td>
</tr>
<tr>
<td>SCALS2</td>
<td>10</td>
<td>9</td>
<td>5356.7</td>
<td>1096045.6</td>
</tr>
<tr>
<td>SCALS3</td>
<td>10</td>
<td>0</td>
<td>13095.3</td>
<td>N/A</td>
</tr>
<tr>
<td>SCALS4</td>
<td>10</td>
<td>0</td>
<td>18883.5</td>
<td>N/A</td>
</tr>
<tr>
<td>SCALS5</td>
<td>10</td>
<td>0</td>
<td>20993.0</td>
<td>N/A</td>
</tr>
<tr>
<td>SCALP1</td>
<td>10</td>
<td>9</td>
<td>193.3</td>
<td>94242.9</td>
</tr>
<tr>
<td>SCALP2</td>
<td>10</td>
<td>10</td>
<td>189.4</td>
<td>631225.2</td>
</tr>
<tr>
<td>SCALP3</td>
<td>10</td>
<td>3</td>
<td>196.6</td>
<td>882251.0</td>
</tr>
<tr>
<td>SCALP4</td>
<td>10</td>
<td>1</td>
<td>248.5</td>
<td>1594201.0</td>
</tr>
<tr>
<td>SCALP5</td>
<td>10</td>
<td>0</td>
<td>283.7</td>
<td>N/A</td>
</tr>
<tr>
<td>SCALP6</td>
<td>10</td>
<td>0</td>
<td>288.4</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Experimental Results
Experimental Results

- So is it worth using?
Experimental Results

• So is it worth using?

• Coram - assigning adopted children to families. ~ 100’s of agents. Preference lists long and probability of ties high
Experimental Results

- So is it worth using?

- Coram - assigning adopted children to families. ~ 100’s of agents. Preference lists long and probability of ties high

- 21 instances, increasing difficulty. Initial IP could only solve first 6 within 5 minutes, approximation algorithm took less than 2 seconds for each
Future Work
Future Work

• Finding an approximation algorithm with a better performance guarantee than 3/2
Future Work

- Finding an approximation algorithm with a better performance guarantee than 3/2
- Finding a better inapproximability result than 33/29

Approximation Algorithms for Stable Matching Problems; PhD thesis; 2007; H. Yanagisawa
Future Work

• Finding an approximation algorithm with a better performance guarantee than 3/2

• Finding a better inapproximability result than 33/29

• coalitions:

Approximation Algorithms for Stable Matching Problems; PhD thesis; 2007; H. Yanagisawa
Future Work

- Finding an approximation algorithm with a better performance guarantee than 3/2

- Finding a better inapproximability result than 33/29

- coalitions:
  - group of several students and lecturers

Approximation Algorithms for Stable Matching Problems; PhD thesis; 2007; H. Yanagisawa
Future Work

• Finding an approximation algorithm with a better performance guarantee than 3/2

• Finding a better inapproximability result than 33/29

• coalitions:
  • group of several students and lecturers
  • permute their assignments

Approximation Algorithms for Stable Matching Problems; PhD thesis; 2007; H. Yanagisawa
Future Work

• Finding an approximation algorithm with a better performance guarantee than $3/2$

• Finding a better inapproximability result than $33/29$

• coalitions:
  • group of several students and lecturers
  • permute their assignments
  • some or all get a better outcome

Approximation Algorithms for Stable Matching Problems; PhD thesis; 2007; H. Yanagisawa
Thank you

Summary

• Student-project allocation problem

• Finding a maximum stable matching
  • Integer programming
  • Approximation algorithm

• Future work: improved performance guarantee; improved inapproximability result; coalitions