An Integer Programming Formulation for a Matching Problem

David Manlove, Duncan Milne and Sofiat Olaosebikan

School of Computing Science, University of Glasgow

BCTCS 2018, Royal Holloway, University of London

March 28, 2018
Outline

1 Introduction
   - Matching Problems
   - Student-Project Allocation problem (SPA)
   - SPA with preferences over Projects (SPA-P)
   - The problem: MAX-SPA-P

2 An Integer Programming (IP) model for MAX-SPA-P

3 Experimental results

4 Discussions and Future work
This class of problem generally involves
- assigning a set of agents to another set of agents
This class of problem generally involves

- assigning a set of agents to another set of agents
- based on the preferences of the agents
Matching Problems

This class of problem generally involves

- assigning a set of agents to another set of agents
- based on the preferences of the agents
- and some problem-specific constraints
This class of problem generally involves
- assigning a set of agents to another set of agents
- based on the preferences of the agents
- and some problem-specific constraints
  - for example, the capacity of the agents
Matching Problems

This class of problem generally involves

- assigning a set of agents to another set of agents
- based on the preferences of the agents
- and some problem-specific constraints
  - for example, the capacity of the agents

Example applications include

- allocation of junior doctors to hospitals
Matching Problems

This class of problem generally involves

- assigning a set of agents to another set of agents
- based on the preferences of the agents
- and some problem-specific constraints
  - for example, the capacity of the agents

Example applications include

- allocation of junior doctors to hospitals
- assigning conference papers to reviewers
This class of problem generally involves

- assigning a set of agents to another set of agents
- based on the preferences of the agents
- and some problem-specific constraints
  - for example, the capacity of the agents

Example applications include

- allocation of junior doctors to hospitals
- assigning conference papers to reviewers
- assigning students to projects
Student-Project Allocation Problem (SPA)

SPA involves

- the assignment of students to projects offered by lecturers
SPA involves

- the assignment of students to projects offered by lecturers
- based on the capacities of projects and lecturers
SPA involves

- the assignment of students to projects offered by lecturers
- based on the capacities of projects and lecturers
- students’ preferences over projects
Student-Project Allocation Problem (SPA)

SPA involves

- the assignment of students to projects offered by lecturers
- based on the capacities of projects and lecturers
- students’ preferences over projects
- lecturers’ preferences over
  - students (SPA-S), or
Student-Project Allocation Problem (SPA)

SPA involves
- the assignment of students to projects offered by lecturers
- based on the capacities of projects and lecturers
- students’ preferences over projects
- lecturers’ preferences over
  - students (SPA-S), or
  - projects (SPA-P), or
Student-Project Allocation Problem (SPA)

SPA involves

- the assignment of students to projects offered by lecturers
- based on the capacities of projects and lecturers
- students’ preferences over projects
- lecturers’ preferences over
  - students (SPA-S), or
  - projects (SPA-P), or
  - student-project pairs (SPA-(S,P))
SPA involves

- the assignment of students to projects offered by lecturers
- based on the capacities of projects and lecturers
- students’ preferences over projects
- lecturers’ preferences over
  - students (SPA-S), or
  - projects (SPA-P), or
  - student-project pairs (SPA-(S,P))
### SPA with preferences over Projects (SPA-P)

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$.
Lecturer capacities: $d_1 = 2$, $d_2 = 1$. 

---

D. Manlove, D. Milne, S. Olaosebikan

Integer Programming
SPA with preferences over Projects (SPA-P)

### Students’ preferences

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>p3</td>
<td>p2</td>
<td>p1</td>
</tr>
<tr>
<td>2</td>
<td>p1</td>
<td>p2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>p3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Lecturers’ preferences

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>p1</td>
<td>p2</td>
</tr>
<tr>
<td>2</td>
<td>p3</td>
<td></td>
</tr>
</tbody>
</table>

### Project capacities

$c_1 = c_2 = c_3 = 1.$

### Lecturer capacities

$d_1 = 2$, $d_2 = 1$.

---

**What we seek...**

- a *matching* of students to projects based on these preferences
SPA with preferences over Projects (SPA-P)

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$.  
Lecturer capacities: $d_1 = 2$, $d_2 = 1$.

What we seek...
- a *matching* of students to projects based on these preferences
  - each student is not assigned more than one project
### SPA with preferences over Projects (SPA-P)

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$.

Lecturer capacities: $d_1 = 2$, $d_2 = 1$.

---

**What we seek...**

- a *matching* of students to projects based on these preferences
  - each student is not assigned more than one project
  - capacities of projects and lecturers are not exceeded
A matching..

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$.  
Lecturer capacities: $d_1 = 2$, $d_2 = 1$.  

D. Manlove, D. Milne, S. Olaosebikan  
Integer Programming  
BCTCS 2018  
6 / 26
A matching..

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$.
Lecturer capacities: $d_1 = 2$, $d_2 = 1$.

however,

- $s_2$ would prefer to be assigned $p_1$
A matching..

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$. Lecturer capacities: $d_1 = 2$, $d_2 = 1$.

however,

- $s_2$ would prefer to be assigned $p_1$
- this means $l_1$ also gets her most preferred project
A matching..

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$.
Lecturer capacities: $d_1 = 2$, $d_2 = 1$.

however,

- $s_2$ would prefer to be assigned $p_1$
- this means $l_1$ also gets her most preferred project
- we call $(s_2, p_1)$ a blocking pair
Definition: Blocking Pair

Given an instance $I$ of SPA-P, and a matching $M$ in $I$. The pair $(s_i, p_j)$ forms a blocking pair relative to $M$, where $l_k$ is the lecturer who offers $p_j$, if:

1. either $s_i$ is unassigned in $M$ or $s_i$ prefers $p_j$ to $M$ ($s_i$), and
2. $p_j$ is undersubscribed in $M$, and either (i) $s_i \in M(l_k)$ and $l_k$ prefers $p_j$ to $M(s_i)$, or (ii) $s_i \not\in M(l_k)$ and $l_k$ is undersubscribed, or (iii) $s_i \not\in M(l_k)$ and $l_k$ prefers $p_j$ to her worst non-empty project in $M(l_k)$.
Definition: Blocking Pair

Given an instance $I$ of SPA-P, and a matching $M$ in $I$. The pair $(s_i, p_j)$ forms a *blocking pair* relative to $M$, where $l_k$ is the lecturer who offers $p_j$, if:

1. either $s_i$ is unassigned in $M$ or $s_i$ prefers $p_j$ to $M$ ($s_i$);
2. $p_j$ is undersubscribed in $M$, and either
   - (i) $s_i \in M(l_k)$ and $l_k$ prefers $p_j$ to $M(s_i)$, or
   - (ii) $s_i \not\in M(l_k)$ and $l_k$ is undersubscribed, or
   - (iii) $s_i \not\in M(l_k)$ and $l_k$ prefers $p_j$ to her worst non-empty project in $M(l_k)$.
Definition: Blocking Pair

Given an instance $I$ of SPA-P, and a matching $M$ in $I$. The pair $(s_i, p_j)$ forms a blocking pair relative to $M$, where $l_k$ is the lecturer who offers $p_j$, if:

1. either $s_i$ is unassigned in $M$ or $s_i$ prefers $p_j$ to $M(s_i)$, and
Definition: Blocking Pair

Given an instance $I$ of SPA-P, and a matching $M$ in $I$. The pair $(s_i, p_j)$ forms a blocking pair relative to $M$, where $l_k$ is the lecturer who offers $p_j$, if:

1. either $s_i$ is unassigned in $M$ or $s_i$ prefers $p_j$ to $M(s_i)$, and
2. $p_j$ is undersubscribed in $M$, and either
Definition: Blocking Pair

Given an instance $I$ of SPA-P, and a matching $M$ in $I$. The pair $(s_i, p_j)$ forms a blocking pair relative to $M$, where $l_k$ is the lecturer who offers $p_j$, if:

1. either $s_i$ is unassigned in $M$ or $s_i$ prefers $p_j$ to $M(s_i)$, and
2. $p_j$ is undersubscribed in $M$, and either
   (i) $s_i \in M(l_k)$ and $l_k$ prefers $p_j$ to $M(s_i)$, or
Definition: Blocking Pair

Given an instance $I$ of SPA-P, and a matching $M$ in $I$. The pair $(s_i, p_j)$ forms a blocking pair relative to $M$, where $l_k$ is the lecturer who offers $p_j$, if:

1. either $s_i$ is unassigned in $M$ or $s_i$ prefers $p_j$ to $M(s_i)$, and
2. $p_j$ is undersubscribed in $M$, and either
   (i) $s_i \in M(l_k)$ and $l_k$ prefers $p_j$ to $M(s_i)$, or
   (ii) $s_i \notin M(l_k)$ and $l_k$ is undersubscribed, or
Definition: Blocking Pair

Given an instance $I$ of SPA-P, and a matching $M$ in $I$. The pair $(s_i, p_j)$ forms a *blocking pair* relative to $M$, where $l_k$ is the lecturer who offers $p_j$, if:

1. either $s_i$ is unassigned in $M$ or $s_i$ prefers $p_j$ to $M(s_i)$, and
2. $p_j$ is undersubscribed in $M$, and either
   (i) $s_i \in M(l_k)$ and $l_k$ prefers $p_j$ to $M(s_i)$, or
   (ii) $s_i \notin M(l_k)$ and $l_k$ is undersubscribed, or
   (iii) $s_i \notin M(l_k)$ and $l_k$ prefers $p_j$ to her worst non-empty project in $M(l_k)$. 

D. Manlove, D. Milne, S. Olaosebikan (School of Computing Science, University of Glasgow)
Students' preferences

\begin{align*}
  s_1: & \quad p_3 \quad p_2 \quad p_1 \\
  s_2: & \quad p_1 \quad p_2 \\
  s_3: & \quad p_3
\end{align*}

Lecturers' preferences

\begin{align*}
  l_1: & \quad p_1 \quad p_2 \\
  l_2: & \quad p_3
\end{align*}

Project capacities: \( c_1 = c_2 = c_3 = 1 \).

Lecturer capacities: \( d_1 = 2 \), \( d_2 = 1 \).
Another matching..

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$.
Lecturer capacities: $d_1 = 2$, $d_2 = 1$.

- $s_1$ and $s_2$ would rather swap their assigned projects, in order to be better off.
Another matching..

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1 ): ( p_3 ) ( p_2 ) ( p_1 )</td>
<td>( l_1 ): ( p_1 ) ( p_2 )</td>
</tr>
<tr>
<td>( s_2 ): ( p_1 ) ( p_2 )</td>
<td>( l_2 ): ( p_3 )</td>
</tr>
<tr>
<td>( s_3 ): ( p_3 )</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: \( c_1 = c_2 = c_3 = 1 \).
Lecturer capacities: \( d_1 = 2 \), \( d_2 = 1 \).

- \( s_1 \) and \( s_2 \) would rather swap their assigned projects, in order to be better off.
- we call \( \{ s_1, s_2 \} \) a coalition.
Definition: Coalition

Given a matching $M$, a *coalition* is a set of students $\{s_{i_0}, \ldots, s_{i_{r-1}}\}$, for some $r \geq 2$ such that each student $s_{i_j}$ ($0 \leq j \leq r - 1$) is assigned in $M$ and prefers $M(s_{i_{j+1}})$ to $M(s_{i_j})$, where addition is performed modulo $r$. 
Given a matching $M$, a *coalition* is a set of students ${s_{i_0}, \ldots, s_{i_{r-1}}}$, for some $r \geq 2$ such that each student $s_{i_j}$ ($0 \leq j \leq r - 1$) is assigned in $M$ and prefers $M(s_{i_{j+1}})$ to $M(s_{i_j})$, where addition is performed modulo $r$. 
Definition: Coalition

Given a matching $M$, a *coalition* is a set of students $\{s_{i_0}, \ldots, s_{i_{r-1}}\}$, for some $r \geq 2$ such that each student $s_{i_j}$ ($0 \leq j \leq r - 1$) is assigned in $M$ and prefers $M(s_{i_{j+1}})$ to $M(s_{i_j})$, where addition is performed modulo $r$. 
Definition: Coalition

Given a matching $M$, a *coalition* is a set of students $\{s_{i_0}, \ldots, s_{i_{r-1}}\}$, for some $r \geq 2$ such that each student $s_{i_j}$ ($0 \leq j \leq r - 1$) is assigned in $M$ and prefers $M(s_{i_{j+1}})$ to $M(s_{i_j})$, where addition is performed modulo $r$. 
Given a matching $M$, a coalition is a set of students $\{s_{i_0}, \ldots, s_{i_{r-1}}\}$, for some $r \geq 2$ such that each student $s_{i_j}$ ($0 \leq j \leq r - 1$) is assigned in $M$ and prefers $M(s_{i_{j+1}})$ to $M(s_{i_j})$, where addition is performed modulo $r$. 
Definition: Coalition

Given a matching $M$, a *coalition* is a set of students $\{s_{i_0}, \ldots, s_{i_{r-1}}\}$, for some $r \geq 2$ such that each student $s_{ij}$ ($0 \leq j \leq r - 1$) is assigned in $M$ and prefers $M(s_{ij}+1)$ to $M(s_{ij})$, where addition is performed modulo $r$. 
Definition: Coalition

Given a matching $M$, a coalition is a set of students $\{s_{i_0}, \ldots, s_{i_{r-1}}\}$, for some $r \geq 2$ such that each student $s_{i_j}$ ($0 \leq j \leq r - 1$) is assigned in $M$ and prefers $M(s_{i_{j+1}})$ to $M(s_{i_j})$, where addition is performed modulo $r$. 
The type of matching we seek..
The type of matching we seek..

Stable matchings
- one with no blocking pair and no coalition

Stable matchings.

A stable matching

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

- 2 students are matched
### A stable matching

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$  $p_2$  $p_1$</td>
<td>$l_1$: $p_1$  $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$  $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

- 2 students are matched

### Another stable matching

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$  $p_2$  $p_1$</td>
<td>$l_1$: $p_1$  $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$  $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

- 3 students are matched
Maximum cardinality stable matching

Another problem..

- finding a maximum cardinality stable matching (MAX-SPA-P)
Another problem..

- finding a maximum cardinality stable matching (MAX-SPA-P)
- MAX-SPA-P is NP-hard
Another problem...

- finding a maximum cardinality stable matching (MAX-SPA-P)
- MAX-SPA-P is NP-hard

Existing results for MAX-SPA-P
Maximum cardinality stable matching

Another problem..

- finding a maximum cardinality stable matching (MAX-SPA-P)
- MAX-SPA-P is NP-hard

Existing results for MAX-SPA-P

Suppose the size of a maximum stable matching $M$ is 12,

- 2-approximation algorithm\(^a\), i.e., solution at least $\frac{1}{2}M = 6$


Maximum cardinality stable matching

Another problem..

- finding a maximum cardinality stable matching (MAX-SPA-P)
- MAX-SPA-P is NP-hard

Existing results for MAX-SPA-P

Suppose the size of a maximum stable matching $M$ is 12,

- 2-approximation algorithm\(^a\), i.e., solution at least $\frac{1}{2}M = 6$
- $\frac{3}{2}$-approximation algorithm\(^b\), i.e., solution at least $\frac{2}{3}M = 8$
- not approximable within $\frac{21}{19} - \epsilon$, for any $\epsilon > 0$, unless P = NP


An Integer Programming (IP) model for MAX-SPA-P

A general construction of our IP model
create binary-valued variables to represent the assignment of students to projects;
enforce the following classes of constraints:
1. find a matching;
2. ensure matching does not admit a blocking pair;
3. ensure matching does not admit a coalition;
describe an objective function to maximise the size of the matching.
An Integer Programming (IP) model for MAX-SPA-P

A general construction of our IP model

1. find a matching;
2. ensure matching does not admit a blocking pair;
3. ensure matching does not admit a coalition;
4. describe an objective function to maximise the size of the matching.
A general construction of our IP model

- create binary-valued variables to represent the assignment of students to projects;
An Integer Programming (IP) model for MAX-SPA-P

A general construction of our IP model

- create binary-valued variables to represent the assignment of students to projects;
- enforce the following classes of constraints:
A general construction of our IP model

- create binary-valued variables to represent the assignment of students to projects;
- enforce the following classes of constraints:
  1. find a matching;
A general construction of our IP model

- create binary-valued variables to represent the assignment of students to projects;
- enforce the following classes of constraints:
  1. find a matching;
  2. ensure matching does not admit a blocking pair;
A general construction of our IP model

- create binary-valued variables to represent the assignment of students to projects;
- enforce the following classes of constraints:
  1. find a matching;
  2. ensure matching does not admit a blocking pair;
  3. ensure matching does not admit a coalition;
A general construction of our IP model

- create binary-valued variables to represent the assignment of students to projects;
- enforce the following classes of constraints:
  1. find a matching;
  2. ensure matching does not admit a blocking pair;
  3. ensure matching does not admit a coalition;
- describe an objective function to maximise the size of the matching.
Encoding the binary-valued variables

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$.
Lecturer capacities: $d_1 = 2$, $d_2 = 1$.

We encode each $(s_i, p_j)$ as a variable $x_{i,j} \in \{0, 1\}$
### Encoding the binary-valued variables

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$.
Lecturer capacities: $d_1 = 2$, $d_2 = 1$.

We encode each $(s_i, p_j)$ as a variable $x_{i,j} \in \{0, 1\}$

$x_{1,3}$ $x_{1,2}$ $x_{1,1}$
### Encoding the binary-valued variables

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$.
Lecturer capacities: $d_1 = 2$, $d_2 = 1$.

We encode each $(s_i, p_j)$ as a variable $x_{i,j} \in \{0, 1\}$

\[
x_{1,3} \quad x_{1,2} \quad x_{1,1} \quad \downarrow \quad 1, \text{ then } s_1 \text{ is assigned to } p_3
\]
Encoding the binary-valued variables

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$.
Lecturer capacities: $d_1 = 2$, $d_2 = 1$.

We encode each $(s_i, p_j)$ as a variable $x_{i,j} \in \{0, 1\}$

\[
x_{1,3} \quad x_{1,2} \quad x_{1,1}
\]

\[= 1, \text{ then } s_1 \text{ is assigned to } p_3
\]

\[= 0, \text{ then } s_1 \text{ is not assigned to } p_3
\]
## Encoding the binary-valued variables

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1: \ p_3 \ p_2 \ p_1 )</td>
<td>( l_1: \ p_1 \ p_2 )</td>
</tr>
<tr>
<td>( s_2: \ p_1 \ p_2 )</td>
<td>( l_2: \ p_3 )</td>
</tr>
<tr>
<td>( s_3: \ p_3 )</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: \( c_1 = c_2 = c_3 = 1 \).
Lecturer capacities: \( d_1 = 2, \ d_2 = 1 \).

We encode each \((s_i, p_j)\) as a variable \(x_{i,j} \in \{0, 1\}\)

\[
\begin{align*}
x_{1,3} & \quad x_{1,2} & \quad x_{1,1} \\
\downarrow & & & \Downarrow \\
& = 1, \text{ then } s_1 \text{ is assigned to } p_3 \\
& = 0, \text{ then } s_1 \text{ is not assigned to } p_3
\end{align*}
\]

\( x_{2,1} \quad x_{2,2} \)
## Encoding the binary-valued variables

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$.
Lecturer capacities: $d_1 = 2$, $d_2 = 1$.

We encode each $(s_i, p_j)$ as a variable $x_{i,j} \in \{0, 1\}$

$x_{1,3} \quad x_{1,2} \quad x_{1,1}$

$\Downarrow$

- $= 1$, then $s_1$ is assigned to $p_3$
- $= 0$, then $s_1$ is not assigned to $p_3$

$x_{2,1} \quad x_{2,2}$

$x_{3,3}$
## Matching Constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$  $p_2$  $p_1$</td>
<td>$l_1$: $p_1$  $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$  $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

- Project capacities: $c_1 = c_2 = c_3 = 1$.
- Lecturer capacities: $d_1 = 2$, $d_2 = 1$. 

---

D. Manlove, D. Milne, S. Olaosebikan

Integer Programming
Matching Constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$.
Lecturer capacities: $d_1 = 2$, $d_2 = 1$.

- each student is not assigned more than one project
### Matching Constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$.
Lecturer capacities: $d_1 = 2$, $d_2 = 1$.

- each student is not assigned more than one project

\[
\sum_{p_j \in A_i} x_{i,j} \leq 1 \quad (1 \leq i \leq n_1), \quad \Rightarrow
\]
Matching Constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1: \ p_3 \ p_2 \ p_1 )</td>
<td>( l_1: \ p_1 \ p_2 )</td>
</tr>
<tr>
<td>( s_2: \ p_1 \ p_2 )</td>
<td>( l_2: \ p_3 )</td>
</tr>
<tr>
<td>( s_3: \ p_3 )</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: \( c_1 = c_2 = c_3 = 1 \).
Lecturer capacities: \( d_1 = 2, \ d_2 = 1 \).

- each student is not assigned more than one project

\[
\sum_{p_j \in A_i} x_{i,j} \leq 1 \ (1 \leq i \leq n_1), \quad \implies x_{1,3} + x_{1,2} + x_{1,1} \leq 1
\]
### Matching Constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

- Project capacities: $c_1 = c_2 = c_3 = 1$.
- Lecturer capacities: $d_1 = 2$, $d_2 = 1$.

- each student is not assigned more than one project

\[ \sum_{p_j \in A_i} x_{i,j} \leq 1 \quad (1 \leq i \leq n_1), \quad \implies x_{1,3} + x_{1,2} + x_{1,1} \leq 1 \]

- capacities of projects are not exceeded
### Matching Constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1: \ p_3 \ p_2 \ p_1 )</td>
<td>( l_1: \ p_1 \ p_2 )</td>
</tr>
<tr>
<td>( s_2: \ p_1 \ p_2 )</td>
<td>( l_2: \ p_3 )</td>
</tr>
<tr>
<td>( s_3: \ p_3 )</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: \( c_1 = c_2 = c_3 = 1 \).
Lecturer capacities: \( d_1 = 2, \ d_2 = 1 \).

- Each student is not assigned more than one project
  \[
  \sum_{p_j \in A_i} x_{i,j} \leq 1 \quad (1 \leq i \leq n_1), \quad \Rightarrow \quad x_{1,3} + x_{1,2} + x_{1,1} \leq 1
  \]

- Capacities of projects are not exceeded
  \[
  \sum_{i=1}^{n_1} x_{i,j} \leq c_j, \quad (1 \leq j \leq n_2)
  \]
Matching Constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>s₁: p₃ p₂ p₁</td>
<td>l₁: p₁ p₂</td>
</tr>
<tr>
<td>s₂: p₁ p₂</td>
<td>l₂: p₃</td>
</tr>
<tr>
<td>s₃: p₃</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: \( c₁ = c₂ = c₃ = 1 \).
Lecturer capacities: \( d₁ = 2, \ d₂ = 1 \).

- each student is not assigned more than one project

\[
\sum_{p_j \in A_i} x_{i,j} \leq 1 \quad (1 \leq i \leq n₁), \quad \Rightarrow \quad x_{1,3} + x_{1,2} + x_{1,1} \leq 1
\]

- capacities of projects are not exceeded

\[
\sum_{i=1}^{n₁} x_{i,j} \leq c_j, \quad (1 \leq j \leq n₂) \quad \Rightarrow \quad x_{1,1} + x_{2,1} \leq 1
\]
Matching Constraints..

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$.
Lecturer capacities: $d_1 = 2$, $d_2 = 1$.

- capacities of lecturers are not exceeded
Matching Constraints...

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1.$
Lecturer capacities: $d_1 = 2$, $d_2 = 1.$

- capacities of lecturers are not exceeded

\[
\sum_{i=1}^{n_1} \sum_{p_j \in P_k} x_{i,j} \leq d_k \quad (1 \leq k \leq n_3),
\]
### Matching Constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1: \quad p_3 \quad p_2 \quad p_1 )</td>
<td>( l_1: \quad p_1 \quad p_2 )</td>
</tr>
<tr>
<td>( s_2: \quad p_1 \quad p_2 )</td>
<td>( l_2: \quad p_3 )</td>
</tr>
<tr>
<td>( s_3: \quad p_3 )</td>
<td></td>
</tr>
</tbody>
</table>

#### Project capacities

\[ c_1 = c_2 = c_3 = 1. \]

#### Lecturer capacities

\[ d_1 = 2, \quad d_2 = 1. \]

- capacities of lecturers are not exceeded

\[
\sum_{i=1}^{n_1} \sum_{p_j \in P_k} x_{i,j} \leq d_k \quad (1 \leq k \leq n_3),
\]

\[ \implies x_{1,2} + x_{1,1} + x_{2,1} + x_{2,2} \leq 2 \]
Blocking pair constraints

Students' preferences Lecturers' preferences

\[ s_1: p_3 \quad p_2 \quad p_1 \quad l_1: p_1 \quad p_2 \quad s_2: p_1 \quad p_2 \quad l_2: p_3 \quad s_3: p_3 \]

Project capacities:

\[ c_1 = c_2 = c_3 = 1. \]

Lecturer capacities:

\[ d_1 = 2, \quad d_2 = 1. \]

For each \((s_i, p_j)\), where \(l_k\) is the lecturer who offers \(p_j\), we define \(\theta_{i,j} = 1 - \sum p_j' \in S_{i,j} x_{i,j}' \Rightarrow \theta_{2,1} = 1 - x_{2,1} = 1.\)

Create \(\alpha_j \in \{0, 1\}\), enforce \(c_j \alpha_j \geq c_j - \sum_{i=1}^{n} x_{i,j} \Rightarrow \alpha_1 = 1.\)

Define \(\gamma_{i,j,k} = \sum p_j' \in T_{k,j} x_{i,j}' \); \(T_{1,1} = \{p_2\}\) \(\Rightarrow \gamma_{2,1,1} = x_{2,2} = 1.\)

\( (i) \quad \theta_{i,j} + \alpha_j + \gamma_{i,j,k} \leq 2; \)

\( (ii) \quad \theta_{i,j} + \alpha_j + (1 - \beta_{i,k}) + \delta_k \leq 3; \)

\( (iii) \quad \theta_{i,j} + \alpha_j + (1 - \beta_{i,k}) + \eta_{j,k} \leq 3. \)
Blocking pair constraints

Students’ preferences  
- $s_1$: $p_3$ $p_2$ $p_1$  
- $s_2$: $p_1$ $p_2$  
- $s_3$: $p_3$

Lecturers’ preferences  
- $l_1$: $p_1$ $p_2$  
- $l_2$: $p_3$

Project capacities: $c_1 = c_2 = c_3 = 1$.  
Lecturer capacities: $d_1 = 2$, $d_2 = 1$.  

\[ \theta_{i,j} = 1 - \sum_{p_{j'} \in S_{i,j}} x_{i,j'} \]  
\[ \alpha_j \in \{0, 1\} \]  
\[ \gamma_{i,j,k} = \sum_{p_{j'} \in T_{k,j}} x_{i,j'} \]  
\(T_{1,1} = \{p_2\}\)  

\[ \theta_{i,j} + \alpha_j + \gamma_{i,j,k} \leq 2; \]
\[ \theta_{i,j} + \alpha_j + (1 - \beta_{i,k}) + \delta_k \leq 3; \]
\[ \theta_{i,j} + \alpha_j + (1 - \beta_{i,k}) + \eta_{j,k} \leq 3. \]
## Blocking pair constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s_1): (p_3) (p_2) (p_1)</td>
<td>(l_1): (p_1) (p_2)</td>
</tr>
<tr>
<td>(s_2): (p_1) (p_2)</td>
<td>(l_2): (p_3)</td>
</tr>
<tr>
<td>(s_3): (p_3)</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: \(c_1 = c_2 = c_3 = 1\).

Lecturer capacities: \(d_1 = 2\), \(d_2 = 1\).

For each \((s_i, p_j)\), where \(l_k\) is the lecturer who offers \(p_j\), we

- define \(\theta_{i,j} = 1 - \sum_{p_{j'} \in S_{i,j}} x_{i,j'}\)
Blocking pair constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1: \ p_3 \ p_2 \ p_1 )</td>
<td>( l_1: \ p_1 \ p_2 )</td>
</tr>
<tr>
<td>( s_2: \ p_1 \ p_2 )</td>
<td>( l_2: \ p_3 )</td>
</tr>
<tr>
<td>( s_3: \ p_3 )</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: \( c_1 = c_2 = c_3 = 1 \).
Lecturer capacities: \( d_1 = 2, \ d_2 = 1 \).

For each \((s_i, p_j)\), where \(l_k\) is the lecturer who offers \(p_j\), we

- define \( \theta_{i,j} = 1 - \sum_{p_{j'} \in S_{i,j}} x_{i,j'} \implies \theta_{2,1} = 1 - x_{2,1} = 1. \)
Blocking pair constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$.
Lecturer capacities: $d_1 = 2$, $d_2 = 1$.

For each $(s_i, p_j)$, where $l_k$ is the lecturer who offers $p_j$, we

- define $\theta_{i,j} = 1 - \sum_{p_{j'} \in S_{i,j}} x_{i,j'} \implies \theta_{2,1} = 1 - x_{2,1} = 1$.
- create $\alpha_j \in \{0, 1\}$, enforce $c_j \alpha_j \geq c_j - \sum_{i'=1}^{n_1} x_{i',j}$
Blocking pair constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$. Lecturer capacities: $d_1 = 2$, $d_2 = 1$.

For each $(s_i, p_j)$, where $l_k$ is the lecturer who offers $p_j$, we

- define $\theta_{i,j} = 1 - \sum_{p_{j'} \in S_{i,j}} x_{i,j'}$ $\implies$ $\theta_{2,1} = 1 - x_{2,1} = 1$.
- create $\alpha_j \in \{0, 1\}$, enforce $c_j \alpha_j \geq c_j - \sum_{i'=1}^{n_1} x_{i',j}$ $\implies$ $\alpha_1 = 1$. 
## Blocking pair constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1: p_3 p_2 p_1 )</td>
<td>( l_1: p_1 p_2 )</td>
</tr>
<tr>
<td>( s_2: p_1 p_2 )</td>
<td>( l_2: p_3 )</td>
</tr>
<tr>
<td>( s_3: p_3 )</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: \( c_1 = c_2 = c_3 = 1 \).
Lecturer capacities: \( d_1 = 2, d_2 = 1 \).

For each \((s_i, p_j)\), where \(l_k\) is the lecturer who offers \(p_j\), we

- define \( \theta_{i,j} = 1 - \sum_{p_{j'} \in S_{i,j}} x_{i,j'} \implies \theta_{2,1} = 1 - x_{2,1} = 1 \).
- create \( \alpha_j \in \{0, 1\} \), enforce \( c_j \alpha_j \geq c_j - \sum_{i'=1}^{n_1} x_{i',j} \implies \alpha_1 = 1 \).
- define \( \gamma_{i,j,k} = \sum_{p_{j'} \in T_{k,j}} x_{i,j'} \).
Blocking pair constraints

Students’ preferences

\[s_1: \ p_3 \ p_2 \ p_1\]
\[s_2: \ p_1 \ p_2\]
\[s_3: \ p_3\]

Lecturers’ preferences

\[l_1: \ p_1 \ p_2\]
\[l_2: \ p_3\]

Project capacities: \(c_1 = c_2 = c_3 = 1\).
Lecturer capacities: \(d_1 = 2\), \(d_2 = 1\).

For each \((s_i, p_j)\), where \(l_k\) is the lecturer who offers \(p_j\), we

- define \(\theta_{i,j} = 1 - \sum_{p_j' \in S_{i,j}} x_{i,j'} \implies \theta_{2,1} = 1 - x_{2,1} = 1\).
- create \(\alpha_j \in \{0, 1\}\), enforce \(c_j \alpha_j \geq c_j - \sum_{i'=1}^{n1} x_{i',j} \implies \alpha_1 = 1\).
- define \(\gamma_{i,j,k} = \sum_{p_j' \in T_{k,j}} x_{i,j'}\); \(T_{1,1} = \{p_2\}\).
Blocking pair constraints

Students’ preferences

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>s1:</td>
<td>p3</td>
<td>p2</td>
</tr>
<tr>
<td>s2:</td>
<td>p1</td>
<td>p2</td>
</tr>
<tr>
<td>s3:</td>
<td>p3</td>
<td></td>
</tr>
</tbody>
</table>

Lecturers’ preferences

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>l1:</td>
<td>p1</td>
</tr>
<tr>
<td>l2:</td>
<td>p3</td>
</tr>
</tbody>
</table>

Project capacities: \( c_1 = c_2 = c_3 = 1 \).

Lecturer capacities: \( d_1 = 2, \ d_2 = 1 \).

For each \((s_i, p_j)\), where \(l_k\) is the lecturer who offers \(p_j\), we

- define \( \theta_{i,j} = 1 - \sum_{p_{j'} \in S_{i,j}} x_{i,j'} \implies \theta_{2,1} = 1 - x_{2,1} = 1 \).
- create \( \alpha_j \in \{0, 1\} \), enforce \( c_j \alpha_j \geq c_j - \sum_{i'=1}^{n_1} x_{i',j} \implies \alpha_1 = 1 \).
- define \( \gamma_{i,j,k} = \sum_{p_{j'} \in T_{k,j}} x_{i,j'} \); \( T_{1,1} = \{p_2\} \implies \gamma_{2,1,1} = x_{2,2} = 1 \).
 Blocking pair constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>s₁: p₃ p₂ p₁</td>
<td>l₁: p₁ p₂</td>
</tr>
<tr>
<td>s₂: p₁ p₂</td>
<td>l₂: p₃</td>
</tr>
<tr>
<td>s₃: p₃</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: \( c_1 = c_2 = c_3 = 1 \).
Lecturer capacities: \( d_1 = 2 \), \( d_2 = 1 \).

For each \((s_i, p_j)\), where \(l_k\) is the lecturer who offers \(p_j\), we

- define \( \theta_{i,j} = 1 - \sum_{p_{j'} \in S_{i,j}} x_{i,j'} \implies \theta_{2,1} = 1 - x_{2,1} = 1 \).
- create \( \alpha_j \in \{0, 1\} \), enforce \( c_j \alpha_j \geq c_j - \sum_{i'=1}^{n_1} x_{i',j} \implies \alpha_1 = 1 \).
- define \( \gamma_{i,j,k} = \sum_{p_{j'} \in T_{k,j}} x_{i,j'} \); \( T_{1,1} = \{p_2\} \implies \gamma_{2,1,1} = x_{2,2} = 1 \).

\[(i) \quad \theta_{i,j} + \alpha_j + \gamma_{i,j,k} \leq 2;\]
Blocking pair constraints

Students’ preferences

\( s_1: \quad p_3 \quad p_2 \quad p_1 \)
\( s_2: \quad p_1 \quad p_2 \)
\( s_3: \quad p_3 \)

Lecturers’ preferences

\( l_1: \quad p_1 \quad p_2 \)
\( l_2: \quad p_3 \)

Project capacities: \( c_1 = c_2 = c_3 = 1 \).
Lecturer capacities: \( d_1 = 2, \quad d_2 = 1 \).

For each \((s_i, p_j)\), where \( l_k \) is the lecturer who offers \( p_j \), we

- define \( \theta_{i,j} = 1 - \sum_{p_{j'} \in S_{i,j}} x_{i,j'} \quad \implies \quad \theta_{2,1} = 1 - x_{2,1} = 1 \).
- create \( \alpha_j \in \{0, 1\} \), enforce \( c_j \alpha_j \geq c_j - \sum_{i'=1}^{n_1} x_{i',j} \quad \implies \quad \alpha_1 = 1 \).
- define \( \gamma_{i,j,k} = \sum_{p_{j'} \in T_{k,j}} x_{i,j'}; \quad T_{1,1} = \{p_2\} \quad \implies \quad \gamma_{2,1,1} = x_{2,2} = 1 \).

\[
\begin{align*}
\text{(i)} \quad \theta_{i,j} + \alpha_j + \gamma_{i,j,k} & \leq 2; \\
\text{(ii)} \quad \theta_{i,j} + \alpha_j + (1 - \beta_{i,k}) + \delta_k & \leq 3;
\end{align*}
\]
# Blocking pair constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: p3 p2 p1</td>
<td>$l_1$: p1 p2</td>
</tr>
<tr>
<td>$s_2$: p1 p2</td>
<td>$l_2$: p3</td>
</tr>
<tr>
<td>$s_3$: p3</td>
<td></td>
</tr>
</tbody>
</table>

Project capacities: $c_1 = c_2 = c_3 = 1$.  
Lecturer capacities: $d_1 = 2$, $d_2 = 1$.

For each $(s_i, p_j)$, where $l_k$ is the lecturer who offers $p_j$, we

- define $\theta_{i,j} = 1 - \sum_{p_{j'} \in S_{i,j}} x_{i,j'} \implies \theta_{2,1} = 1 - x_{2,1} = 1$.
- create $\alpha_j \in \{0, 1\}$, enforce $c_j \alpha_j \geq c_j - \sum_{i'=1}^{n_1} x_{i',j} \implies \alpha_1 = 1$.
- define $\gamma_{i,j,k} = \sum_{p_{j'} \in T_{k,j}} x_{i,j'}$; $T_{1,1} = \{p_2\} \implies \gamma_{2,1,1} = x_{2,2} = 1$.

(i) $\theta_{i,j} + \alpha_j + \gamma_{i,j,k} \leq 2$;  
(ii) $\theta_{i,j} + \alpha_j + (1 - \beta_{i,k}) + \delta_k \leq 3$;  
(iii) $\theta_{i,j} + \alpha_j + (1 - \beta_{i,k}) + \eta_{j,k} \leq 3$. 
## Coalition constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1: p_3 \ p_2 \ p_1 )</td>
<td>( l_1: p_1 \ p_2 )</td>
</tr>
<tr>
<td>( s_2: p_1 \ p_2 )</td>
<td>( l_2: p_3 )</td>
</tr>
<tr>
<td>( s_3: p_3 )</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Envy graph</th>
</tr>
</thead>
</table>

\( s_1 \) admits topological ordering = \( \Rightarrow \) it is acyclic = \( \Rightarrow \) no coalition.

For each \((s_i, s_i')\), if \( s_i \) envies \( s_i' \), create \( e_{i,i'} \in \{0, 1\} \) and enforce \( e_{i,i'} + 1 \geq x_{i,j} + x_{i',j} \)

i \neq i'

To hold the label of each vertex in the topological ordering, create an integer-valued variable \( v_i \) and enforce \( v_i < v_{i'} + n_1(1 - e_{i,i'}) \)

\( n_1 \) – number of students.
Coalition constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Envy graph
Coalition constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Envy graph

$G$ admits topological ordering $\Rightarrow$ it is acyclic $\Rightarrow$ no coalition.

For each $(s_i, s_i')$, if $s_i$ envies $s_i'$, create $e_{i,i'} \in \{0, 1\}$ and enforce $e_{i,i'} + 1 \geq x_{i,j} + x_{i',j'}$.

For each vertex in the topological ordering, create an integer-valued variable $v_i$ and enforce $v_i < v_{i'} + n_1(1 - e_{i,i'})$. $n_1$ is the number of students.
Coalition constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$  $p_2$  $p_1$</td>
<td>$l_1$: $p_1$  $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$  $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Envy graph

- $s_3$ envies $s_1$
- $s_1$ envies $s_2$
Coalition constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Envy graph

$E_1,s_1,l_1 = E_2,s_1,l_1 = 0$

For each $(s_i, s_{i}')$, if $s_i$ envies $s_{i}'$, create $e_{i,i'} \in \{0, 1\}$ and enforce $e_{i,i'} + 1 \geq x_{i,j} + x_{i',j}$ if $i \neq i'$

To hold the label of each vertex in the topological ordering, create an integer-valued variable $v_i$ and enforce $v_i < v_{i'} + n_1(1 - e_{i,i'})$ where $n_1$ is the number of students.
Coalition constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Envy graph

An integer-valued variable $v_i$ is created for each vertex to hold the label of each vertex in the topological ordering.
Coalition constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Envy graph

- admits topological ordering $\implies$ it is acyclic $\implies$ no coalition.
Coalition constraints

<table>
<thead>
<tr>
<th>Students’ preferences</th>
<th>Lecturers’ preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$: $p_3$ $p_2$ $p_1$</td>
<td>$l_1$: $p_1$ $p_2$</td>
</tr>
<tr>
<td>$s_2$: $p_1$ $p_2$</td>
<td>$l_2$: $p_3$</td>
</tr>
<tr>
<td>$s_3$: $p_3$</td>
<td></td>
</tr>
</tbody>
</table>

Envy graph

- admits topological ordering $\implies$ it is acyclic $\implies$ no coalition.
- For each $(s_i, s_{i'})$, if $s_i$ envies $s_{i'}$, create $e_{i,i'} \in \{0, 1\}$ and enforce
  - $e_{i,i'} + 1 \geq x_{i,j} + x_{i',j'} \quad i \neq i'$
Coalition constraints

Students’ preferences

\[ s_1: \ p_3 \ p_2 \ p_1 \]
\[ s_2: \ p_1 \ p_2 \]
\[ s_3: \ p_3 \]

Lecturers’ preferences

\[ l_1: \ p_1 \ p_2 \]
\[ l_2: \ p_3 \]

Envy graph

\[ s_3 \rightarrow s_1 \leftarrow s_2 \]

- admits topological ordering \( \implies \) it is acyclic \( \implies \) no coalition.
- For each \((s_i, s_i')\), if \(s_i\) envies \(s_i'\), create \(e_{i,i'} \in \{0, 1\}\) and enforce
  \[ e_{i,i'} + 1 \geq x_{i,j} + x_{i',j'} \quad i \neq i' \]
- to hold the label of each vertex in the topological ordering, create an integer-valued variable \(v_i\) and enforce
  \[ v_i < v_{i'} + n_1(1 - e_{i,i'}) \quad n_1 - \text{number of students.} \]
Objective function

\[
\max_{n} \sum_{i=1}^{n} \sum_{p_j \in A_i} x_{i,j}
\]

It seeks to maximise the number of students assigned to projects.

Theorem

Given an instance \(I\) of spa-p, there exists an IP formulation \(J\) of \(I\) such that an optimal solution in \(J\) corresponds to a maximum stable matching in \(I\), and vice-versa.

D. Manlove, D. Milne, S. Olaosebikan (School of Computing Science, University of Glasgow)
Objective function

- summation of all the $x_{i,j}$ binary variables

\[
\max \sum_{i=1}^{n_1} \sum_{p_j \in A_i} x_{i,j}
\]
Objective function

- summation of all the $x_{i,j}$ binary variables

$$\text{max} \sum_{i=1}^{n_1} \sum_{p_j \in A_i} x_{i,j}$$

- it seeks to maximise the number of students assigned to projects
Objective function

- summation of all the $x_{i,j}$ binary variables

$$\max \sum_{i=1}^{n_1} \sum_{p_j \in A_i} x_{i,j}$$

- it seeks to maximise the number of students assigned to projects

Theorem

Given an instance $I$ of spa-P, there exists an IP formulation $J$ of $I$ such that an optimal solution in $J$ corresponds to a maximum stable matching in $I$, and vice-versa.
Implementation and Experimental Setup

The IP model was implemented using the Gurobi optimisation solver to investigate how the solution produced by the approximation algorithms compares to the optimal solution obtained from the IP model, with respect to the size of the stable matchings constructed.

For this experiment, the IP solver was run on instances involving 1000 students with the coalition constraints removed, resulting in a maximum stable matching size of approximately 63.50 seconds. Without the coalition constraints, the size was approximately 2.61 seconds.

For the purpose of this experiment, we removed the coalition constraints from our IP solver.
IP model was implemented using the Gurobi optimisation solver

- www.gurobi.com
Implementation and Experimental Setup

- IP model was implemented using the Gurobi optimisation solver
  - [www.gurobi.com](http://www.gurobi.com)
- to investigate how the solution produced by the approximation algorithms compares to the optimal solution obtained from the IP model, with respect to the size of the stable matchings constructed
Implementation and Experimental Setup

- IP model was implemented using the Gurobi optimisation solver
  - www.gurobi.com
- to investigate how the solution produced by the approximation algorithms compares to the optimal solution obtained from the IP model, with respect to the size of the stable matchings constructed
- IP solver on instance size involving 1000 students
  - with the coalition constraints (63.50 seconds)
  - without the coalition constraints (2.61 seconds)
Implementation and Experimental Setup

- IP model was implemented using the Gurobi optimisation solver
  - www.gurobi.com
- to investigate how the solution produced by the approximation algorithms compares to the optimal solution obtained from the IP model, with respect to the size of the stable matchings constructed
- IP solver on instance size involving 1000 students
  - with the coalition constraints (63.50 seconds)
  - without the coalition constraints (2.61 seconds)
- size of a maximum stable matching = size of a matching that admits no blocking pair, but potentially admits a coalition
Implementation and Experimental Setup

- IP model was implemented using the Gurobi optimisation solver
  - www.gurobi.com
- to investigate how the solution produced by the approximation algorithms compares to the optimal solution obtained from the IP model, with respect to the size of the stable matchings constructed
- IP solver on instance size involving 1000 students
  - with the coalition constraints (63.50 seconds)
  - without the coalition constraints (2.61 seconds)
- size of a maximum stable matching = size of a matching that admits no blocking pair, but potentially admits a coalition
- for the purpose of this experiment, we removed the coalition constraints from our IP solver
Experimental results: Randomly-generated SPA-P instances

![Graph showing the approximate solution vs number of students.](image-url)
Experimental results: Randomly-generated \textsc{spa-p} instances
Experimental results: SPA-P instances derived from real datasets

<table>
<thead>
<tr>
<th>Year</th>
<th>n₁</th>
<th>n₂</th>
<th>n₃</th>
<th>l</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014</td>
<td>55</td>
<td>149</td>
<td>38</td>
<td>6</td>
</tr>
<tr>
<td>2015</td>
<td>76</td>
<td>197</td>
<td>46</td>
<td>6</td>
</tr>
<tr>
<td>2016</td>
<td>92</td>
<td>214</td>
<td>44</td>
<td>6</td>
</tr>
<tr>
<td>2017</td>
<td>90</td>
<td>289</td>
<td>59</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 1: A, B, C, D and E denotes the solution obtained from the IP model, 100 runs of $3/2$-approximation algorithm, single run of $3/2$-approximation algorithm, 100 runs of $2$-approximation algorithm, and single run of $2$-approximation algorithm respectively. Also, n₁, n₂, n₃, and l is number of students, number of projects, number of lecturers and length of the students' preference lists respectively.
Experimental results: SPA-P instances derived from real datasets

- actual student preference data from previous runs of project allocation in the School of Computing Science, University of Glasgow; lecturer preference data was derived from this information
Experimental results: SPA-P instances derived from real datasets

- Actual student preference data from previous runs of project allocation in the School of Computing Science, University of Glasgow; lecturer preference data was derived from this information

<table>
<thead>
<tr>
<th>Year</th>
<th>$n_1$</th>
<th>$n_2$</th>
<th>$n_3$</th>
<th>$l$</th>
<th>Random</th>
<th>Most popular</th>
<th>Least popular</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$A$</td>
<td>$B$</td>
<td>$C$</td>
</tr>
<tr>
<td>2014</td>
<td>55</td>
<td>149</td>
<td>38</td>
<td>6</td>
<td>55</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>2015</td>
<td>76</td>
<td>197</td>
<td>46</td>
<td>6</td>
<td>76</td>
<td>76</td>
<td>76</td>
</tr>
<tr>
<td>2016</td>
<td>92</td>
<td>214</td>
<td>44</td>
<td>6</td>
<td>84</td>
<td>82</td>
<td>83</td>
</tr>
<tr>
<td>2017</td>
<td>90</td>
<td>289</td>
<td>59</td>
<td>4</td>
<td>89</td>
<td>87</td>
<td>85</td>
</tr>
</tbody>
</table>

Table 1: $A$, $B$, $C$, $D$ and $E$ denotes the solution obtained from the IP model, 100 runs of $\frac{3}{2}$-approximation algorithm, single run of $\frac{3}{2}$-approximation algorithm, 100 runs of 2-approximation algorithm, and single run of 2-approximation algorithm respectively. Also, $n_1$, $n_2$, $n_3$ and $l$ is number of students, number of projects, number of lecturers and length of the students’ preference lists respectively.
Discussions and Conclusions

The approximation algorithms outperform the expected bound for the 3/2-approximation algorithm finds stable matchings that are very close in size to optimal, even on a single run. An IP solver on instance size involving 10,000 students (100 seconds) shows that the IP model can be employed in practice. Potential coalitions can subsequently be dealt with in polynomial-time.

D. Manlove, D. Milne, S. Olaosebikan

Integer Programming

BCTCS 2018
Discussions and Conclusions

- the approximation algorithms outperform the expected bound
- the $\frac{3}{2}$-approximation algorithm finds stable matchings that are very close in size to optimal, even on a single run
Discussions and Conclusions

- The approximation algorithms outperform the expected bound.
- The $\frac{3}{2}$-approximation algorithm finds stable matchings that are very close in size to optimal, even on a single run.
- IP solver on instance size involving 10,000 students (100 seconds).
Discussions and Conclusions

- the approximation algorithms outperform the expected bound
- the $\frac{3}{2}$-approximation algorithm finds stable matchings that are very close in size to optimal, even on a single run
- IP solver on instance size involving 10,000 students (100 seconds)
- IP model can be employed in practice
Discussions and Conclusions

- the approximation algorithms outperform the expected bound
- the $\frac{3}{2}$-approximation algorithm finds stable matchings that are very close in size to optimal, even on a single run
- IP solver on instance size involving 10,000 students (100 seconds)
- IP model can be employed in practice
- potential coalitions can subsequently be dealt with in polynomial-time
Future work

Interesting directions..
Future work

Interesting directions..

- Approximation algorithm with improved bounds?
Future work

Interesting directions..

- Approximation algorithm with improved bounds?
- Fixed-Parameter Tractable (FPT) algorithm for MAX-SPA-P?
Interesting directions..

- Approximation algorithm with improved bounds?
- Fixed-Parameter Tractable (FPT) algorithm for MAX-SPA-P?
  - each project and lecturer has capacity 1
Future work

Interesting directions..

- Approximation algorithm with improved bounds?
- Fixed-Parameter Tractable (FPT) algorithm for MAX-SPA-P?
  - each project and lecturer has capacity 1 ✗
Future work

Interesting directions..

- Approximation algorithm with improved bounds?
- Fixed-Parameter Tractable (FPT) algorithm for MAX-SPA-P?
  - each project and lecturer has capacity 1
  - all preference lists are of bounded length
Future work

Interesting directions..

- Approximation algorithm with improved bounds?
- Fixed-Parameter Tractable (FPT) algorithm for MAX-SPA-P?
  - each project and lecturer has capacity 1
  - all preference lists are of bounded length
Future work

Interesting directions..

- Approximation algorithm with improved bounds?
- Fixed-Parameter Tractable (FPT) algorithm for MAX-SPA-P?
  - each project and lecturer has capacity 1 ✗
  - all preference lists are of bounded length ✗
  - what if there is a constant number of lecturer?
Future work

Interesting directions..

- Approximation algorithm with improved bounds?
- Fixed-Parameter Tractable (FPT) algorithm for MAX-SPA-P?
  - each project and lecturer has capacity 1 ×
  - all preference lists are of bounded length ×
  - what if there is a constant number of lecturer?
  - might be solvable in polynomial-time with one lecturer?
Future work

Interesting directions..

- Approximation algorithm with improved bounds?
- Fixed-Parameter Tractable (FPT) algorithm for MAX-SPA-P?
  - each project and lecturer has capacity 1 \( \times \)
  - all preference lists are of bounded length \( \times \)
  - what if there is a constant number of lecturer?
  - might be solvable in polynomial-time with one lecturer?
  - remains hard to solve with two lecturers, even if each project has capacity 1
Future work

Interesting directions..

- Approximation algorithm with improved bounds?
- Fixed-Parameter Tractable (FPT) algorithm for MAX-SPA-P?
  - each project and lecturer has capacity 1 ×
  - all preference lists are of bounded length ×
  - what if there is a constant number of lecturer?
  - might be solvable in polynomial-time with one lecturer?
  - remains hard to solve with two lecturers, even if each project has capacity 1 ✓
## Future work

### Interesting directions..

- Approximation algorithm with improved bounds?
- Fixed-Parameter Tractable (FPT) algorithm for $\text{MAX-SPA-P}$?
  - each project and lecturer has capacity 1 $\times$
  - all preference lists are of bounded length $\times$
  - what if there is a constant number of lecturer? $\times$
  - might be solvable in polynomial-time with one lecturer?
  - remains hard to solve with two lecturers, even if each project has capacity 1 $\checkmark$
Future work

Interesting directions..

- Approximation algorithm with improved bounds?
- Fixed-Parameter Tractable (FPT) algorithm for MAX-SPA-P?
  - each project and lecturer has capacity 1 ✓
  - all preference lists are of bounded length ×
  - what if there is a constant number of lecturer? ×
  - might be solvable in polynomial-time with one lecturer?
  - remains hard to solve with two lecturers, even if each project has capacity 1 ✓
  - more parameters yet to be explored..

**Corresponding author:** Sofiat Olaosebikan  
**Website:** [www.dcs.gla.ac.uk/~sofiat](http://www.dcs.gla.ac.uk/~sofiat)  
**Email:** s.olaosebikan.1@research.gla.ac.uk

\(^1\) Supported by grant EP/P028306/1 from the Engineering and Physical Sciences Research Council.  
\(^2\) Supported by a College of Science and Engineering Scholarship, University of Glasgow.