Profile-based optimal matchings in the Student/Project Allocation problem


School of Computing Science
University of Glasgow

BCTCS, April 2014
Outline

1. Motivation
   1.1 The Student/Project Allocation problem (SPA)
   1.2 Profile-based and Cost-based optimality criterion

2. Greedy maximum matchings in SPA
   2.1 A network flow model
   2.2 Algorithm GREEDY-MAX-SPA

3. Other extensions
   3.1 Lecturer lower quotas
   3.2 Project lower quotas
1. Motivation
1.1 The Student/Project Allocation problem (SPA)
1.2 Profile-based and Cost-based optimality criterion

2. Greedy maximum matchings in SPA
2.1 A network flow model
2.2 Algorithm GREEDY-MAX-SPA

3. Other extensions
3.1 Lecturer lower quotas
3.2 Project lower quotas
The Student/Project Allocation problem (SPA)

- SPA involves the assignment of students to individual or group projects offered by lecturers.

- Various constraints may be placed on the required matching:
  - Each student can be matched to at most one project.
  - Each project has a maximum number of students it can be assigned.
  - Each lecturer has a maximum number of students she can be assigned.
  - etc.

- Students have preferences over projects while lecturers may have preferences over:
  - projects (SPA-P) or
  - students (SPA-S) or
  - student-project pairs (SPA-(S,P)).

- Lecturer preferences may also be ignored.
The Student/Project Allocation problem (SPA)

- SPA involves the assignment of students to individual or group projects offered by lecturers.

- Various constraints may be placed on the required matching:
  - Each student can be matched to at most one project.
  - Each project has a maximum number of students it can be assigned.
  - Each lecturer has a maximum number of students she can be assigned.
  - etc.

- Students have preferences over projects while lecturers may have preferences over:
  - projects (SPA-P) or
  - students (SPA-S) or
  - student-project pairs (SPA-(S,P)).

- Lecturer preferences may also be ignored.
The Student/Project Allocation problem (SPA)

- SPA involves the assignment of students to individual or group projects offered by lecturers.

- Various constraints may be placed on the required matching:
  - Each student can be matched to at most one project.
  - Each project has a maximum number of students it can be assigned.
  - Each lecturer has a maximum number of students she can be assigned.
  - etc.

- Students have preferences over projects while lecturers may have preferences over:
  - projects (SPA-P) or
  - students (SPA-S) or
  - student-project pairs (SPA-(S,P)).

- Lecturer preferences may also be ignored.
The Student/Project Allocation problem (SPA)

- SPA involves the assignment of students to individual or group projects offered by lecturers.

- Various constraints may be placed on the required matching:
  - Each student can be matched to at most one project.
  - Each project has a maximum number of students it can be assigned.
  - Each lecturer has a maximum number of students she can be assigned.
  - etc.

- Students have preferences over projects while lecturers may have preferences over:
  - projects (SPA-P) or
  - students (SPA-S) or
  - student-project pairs (SPA-(S,P)).

- Lecturer preferences may also be ignored.
The Student/Project Allocation problem (SPA)

Some applications include:
- Assigning students to projects in the School of Computing Science, Mathematics, etc. at Glasgow University.
- Assigning students to elective courses in the School of Medicine.
- Assigning projects to markers in the School of Computing Science.

SPA can also be used to solve the problem of:
- Assigning employees to roles/positions in companies.
- Assigning conference reviewers to submissions.

SPA is a generalisation of the Capacitated House-Allocation problem CHA.
- CHA involves the assignment of a set of indivisible goods among a set of applicants.
- For example assigning applicants to houses.
- Many applicants can be assigned to a single house with each house having an upper quota.
The Student/Project Allocation problem (SPA)

- Some applications include:
  - Assigning students to projects in the School of Computing Science, Mathematics, etc. at Glasgow University.
  - Assigning students to elective courses in the School of Medicine.
  - Assigning projects to markers in the School of Computing Science.

- SPA can also be used to solve the problem of:
  - Assigning employees to roles/positions in companies.
  - Assigning conference reviewers to submissions.

- SPA is a generalisation of the Capacitated House-Allocation problem (CHA).
  - CHA involves the assignment of a set of indivisible goods among a set of applicants.
  - For example assigning applicants to houses.
  - Many applicants can be assigned to a single house with each house having an upper quota.
The Student/Project Allocation problem (SPA)

- Some applications include:
  - Assigning students to projects in the School of Computing Science, Mathematics, etc. at Glasgow University.
  - Assigning students to elective courses in the School of Medicine.
  - Assigning projects to markers in the School of Computing Science.

- SPA can also be used to solve the problem of:
  - Assigning employees to roles/positions in companies.
  - Assigning conference reviewers to submissions.

- SPA is a generalisation of the Capacitated House- Allocation problem CHA.
  - CHA involves the assignment of a set of indivisible goods among a set of applicants.
  - For example assigning applicants to houses.
  - Many applicants can be assigned to a single house with each house having an upper quota.
1. Motivation
1.1 The Student/Project Allocation problem (SPA)
1.2 Profile-based and Cost-based optimality criterion

2. Greedy maximum matchings in SPA
2.1 A network flow model
2.2 Algorithm GREEDY-MAX-SPA

3. Other extensions
3.1 Lecturer lower quotas
3.2 Project lower quotas
Two-sided preferences and stability

- Students rank projects in strict order of preference.

- Lecturers rank students or projects or student-project pairs in strict order of preference.

- Objective is then to find a stable matching:
  - One in which there exists no student-project pair who can improve their situation by becoming matched to each other.
  - This would typically cause the matching to unravel as other students/projects will lose partners.
Two-sided preferences and stability

- Students rank projects in strict order of preference.

- Lecturers rank students or projects or student-project pairs in strict order of preference.

- Objective is then to find a stable matching:
  - One in which there exists no student-project pair who can improve their situation by becoming matched to each other.
  - This would typically cause the matching to unravel as other students/projects will lose partners.
Two-sided preferences and stability

- Students rank projects in strict order of preference.

- Lecturers rank students or projects or student-project pairs in strict order of preference.

- Objective is then to find a **stable** matching:
  - One in which there exists no student-project pair who can improve their situation by becoming matched to each other.

  - This would typically cause the matching to unravel as other students/projects will lose partners.
In SPA applications two-sided preferences have drawbacks
- Unfair advantage for better students in SPA-S.
- MAX SPA-P is NP-hard.
- Tedious creating lecturer preferences in SPA-(S, P).

Dropping stability also often allow larger matchings to be found.

When preferences exist on one side only, stability becomes irrelevant.

This motivates the need for other optimality criteria. We choose to optimize:
- size of the matching, and subject to that,
- the satisfaction of the students.
In SPA applications two-sided preferences have drawbacks

- Unfair advantage for better students in SPA-S.
- MAX SPA-P is NP-hard.
- Tedious creating lecturer preferences in SPA-(S, P).

Dropping stability also often allow larger matchings to be found.

- When preferences exist on one side only, stability becomes irrelevant.
- This motivates the need for other optimality criteria. We choose to optimize:
  - size of the matching, and subject to that,
  - the satisfaction of the students.
One-sided preferences and profile-based optimality

- In SPA applications two-sided preferences have drawbacks
  - Unfair advantage for better students in SPA-S.
  - MAX SPA-P is NP-hard.
  - Tedious creating lecturer preferences in SPA-(S, P).

- Dropping stability also often allow larger matchings to be found.

- When preferences exist on one side only, stability becomes irrelevant.

- This motivates the need for other optimality criteria. We choose to optimize:
  - size of the matching, and subject to that,
  - the satisfaction of the students.
In SPA applications two-sided preferences have drawbacks

- Unfair advantage for better students in SPA-S.
- MAX SPA-P is NP-hard.
- Tedious creating lecturer preferences in SPA-(S, P).

Dropping stability also often allow larger matchings to be found.

When preferences exist on one side only, stability becomes irrelevant.

This motivates the need for other optimality criteria. We choose to optimize:

- size of the matching, and subject to that,
- the satisfaction of the students.
Profile-based and Cost-based optimality criterion

students' preferences:

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

\[ M_1 = \{(s_1, p_1), (s_2, p_2)\} \]
\[ \rho(M_1) = (2, 0) \]
\[ \lambda(M_1) = 2 \]

students' preferences:

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

\[ M_2 = \{(s_1, p_3), (s_2, p_1), (s_3, p_2)\} \]
\[ \rho(M_2) = (1, 2) \]
\[ \lambda(M_2) = 5 \]

- The profile \( \rho(M) \) is an \( R \)-tuple \( (x_1, x_2, \ldots, x_R) \) where, for each \( r \) (1 \( \leq r \leq R \)), \( x_r = |\{(s_i, p_j) \in M : \text{rank}(s_i, p_j) = r\}| \).

- The cost \( \lambda(M) = \sum_{(s_i, p_j) \in M} \text{rank}(s_i, p_j) \).
Profile-based and Cost-based optimality criterion

students' preferences:

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

\[ M_1 = \{(s_1, p_1), (s_2, p_2)\} \]
\[ \rho(M_1) = (2, 0) \]
\[ \lambda(M_1) = 2 \]

students' preferences:

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

\[ M_2 = \{(s_1, p_3), (s_2, p_1), (s_3, p_2)\} \]
\[ \rho(M_2) = (1, 2) \]
\[ \lambda(M_2) = 5 \]

- The profile \( \rho(M) \) is an \( R \)-tuple \((x_1, x_2, \ldots, x_R)\) where, for each \( r \) \((1 \leq r \leq R)\), \( x_r = |\{(s_i, p_j) \in M : rank(s_i, p_j) = r\}| \).

- The cost \( \lambda(M) = \sum_{(s_i, p_j) \in M} rank(s_i, p_j) \).
students' preferences:

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

\[ M_1 = \{(s_1, p_1), (s_2, p_2)\} \]
\[ \rho(M_1) = (2, 0) \]
\[ \lambda(M_1) = 2 \]

students' preferences:

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

\[ M_2 = \{(s_1, p_3), (s_2, p_1), (s_3, p_2)\} \]
\[ \rho(M_2) = (1, 2) \]
\[ \lambda(M_2) = 5 \]

- The profile \( \rho(M) \) is an \( R \)-tuple \((x_1, x_2, \ldots, x_R)\) where, for each \( r \) \((1 \leq r \leq R)\), \( x_r = |\{(s_i, p_j) \in M : \text{rank}(s_i, p_j) = r\}| \).

- The cost \( \lambda(M) = \sum_{(s_i, p_j) \in M} \text{rank}(s_i, p_j) \).
students' preferences:

\[ s_1 : p_1 \quad p_3 \]
\[ s_2 : p_2 \quad p_1 \]
\[ s_3 : p_2 \]
\[ M_1 = \{(s_1, p_1), (s_2, p_2)\} \]
\[ \rho(M_1) = (2, \ 0) \]
\[ \lambda(M_1) = 2 \]

students' preferences:

\[ s_1 : p_1 \quad p_3 \]
\[ s_2 : p_2 \quad p_1 \]
\[ s_3 : p_2 \]
\[ M_2 = \{(s_1, p_3), (s_2, p_1), (s_3, p_2)\} \]
\[ \rho(M_2) = (1, \ 2) \]
\[ \lambda(M_2) = 5 \]

- The profile \( \rho(M) \) is an \( R \)-tuple \((x_1, x_2, \ldots, x_R)\) where, for each \( r \) \( (1 \leq r \leq R) \), \( x_r = |\{(s_i, p_j) \in M : rank(s_i, p_j) = r\}|. \)

- The cost \( \lambda(M) = \sum_{(s_i, p_j) \in M} rank(s_i, p_j). \)
Profile-based and Cost-based optimality criterion

**Definition**

A rank maximal matching is a matching that has a lexicographically maximum profile.

**Definition**

A greedy maximum matching is a maximum matching that has a lexicographically maximum profile.

**Definition**

A generous maximum matching is a maximum matching that has a lexicographically minimum reverse-profile.

**Definition**

A minimum cost maximum matching is a maximum matching that has minimum cost.
Profile-based and Cost-based optimality criterion

**Definition**

A **rank maximal matching** is a matching that has a lexicographically maximum profile.

**Definition**

A **greedy maximum matching** is a maximum matching that has a lexicographically maximum profile.

**Definition**

A **generous maximum matching** is a maximum matching that has a lexicographically minimum reverse-profile.

**Definition**

A **minimum cost maximum matching** is a maximum matching that has minimum cost.
Profile-based and Cost-based optimality criterion

**Definition**

A **rank maximal matching** is a matching that has a lexicographically maximum profile.

**Definition**

A **greedy maximum matching** is a maximum matching that has a lexicographically maximum profile.

**Definition**

A **generous maximum matching** is a maximum matching that has a lexicographically minimum reverse-profile.

**Definition**

A **minimum cost maximum matching** is a maximum matching that has minimum cost.
Profile-based and Cost-based optimality criterion

**Definition**

A **rank maximal matching** is a matching that has a lexicographically maximum profile.

**Definition**

A **greedy maximum matching** is a maximum matching that has a lexicographically maximum profile.

**Definition**

A **generous maximum matching** is a maximum matching that has a lexicographically minimum reverse-profile.

**Definition**

A **minimum cost maximum matching** is a maximum matching that has minimum cost.
Profile-based and Cost-based optimality criterion

students' preferences:

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

\[ M_1 = \{(s_1, p_1), (s_2, p_2)\} \]
\[ \rho(M_1) = (2, \ 0) \]
\[ \lambda(M) = 2 \]

\[ \quad \]

students' preferences:

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

\[ M_2 = \{(s_1, p_3), (s_2, p_1), (s_3, p_2)\} \]
\[ \rho(M_2) = (1, \ 2) \]
\[ \lambda(M) = 5 \]

- A rank maximal matching need not be of maximum cardinality.
Profile-based and Cost-based optimality criterion

students' preferences

\[ s_1 : p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \]
\[ s_2 : p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \]
\[ s_3 : p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \]
\[ s_4 : p_1 \quad p_3 \quad p_5 \quad p_4 \quad p_2 \]
\[ s_5 : p_2 \quad p_5 \quad p_4 \quad p_3 \quad p_1 \]

\[ M_1 = \{(s_1, p_1), (s_2, p_4), (s_3, p_5), (s_4, p_3), (s_5, p_2)\} \]
\[ M_2 = \{(s_1, p_1), (s_2, p_2), (s_3, p_3), (s_4, p_5), (s_5, p_4)\} \]
\[ M_3 = \{(s_1, p_1), (s_2, p_2), (s_3, p_4), (s_4, p_3), (s_5, p_5)\} \]

<table>
<thead>
<tr>
<th>Matching ( M_i )</th>
<th>Profile ( \rho(M_i) )</th>
<th>Cost ( \lambda(M_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy Maximum ( M_1 )</td>
<td>(2, 1, 0, 1, 1)</td>
<td>13</td>
</tr>
<tr>
<td>Generous Maximum ( M_2 )</td>
<td>(1, 1, 3, 0, 0)</td>
<td>12</td>
</tr>
<tr>
<td>Minimum Cost Maximum ( M_3 )</td>
<td>(1, 3, 0, 1, 0)</td>
<td>11</td>
</tr>
</tbody>
</table>
Profile-based and Cost-based optimality criterion

students' preferences

\[ s_1 : p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \]
\[ s_2 : p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \]
\[ s_3 : p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \]
\[ s_4 : p_1 \quad p_3 \quad p_5 \quad p_4 \quad p_2 \]
\[ s_5 : p_2 \quad p_5 \quad p_4 \quad p_3 \quad p_1 \]

\[ M_1 = \{(s_1, p_1), (s_2, p_4), (s_3, p_5), (s_4, p_3), (s_5, p_2)\} \]
\[ M_2 = \{(s_1, p_1), (s_2, p_2), (s_3, p_3), (s_4, p_5), (s_5, p_4)\} \]
\[ M_3 = \{(s_1, p_1), (s_2, p_2), (s_3, p_4), (s_4, p_3), (s_5, p_5)\} \]

<table>
<thead>
<tr>
<th>Matching  ( M_i )</th>
<th>Profile ( \rho(M_i) )</th>
<th>Cost ( \lambda(M_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy Maximum ( M_1 )</td>
<td>(2, 1, 0, 1, 1)</td>
<td>13</td>
</tr>
<tr>
<td>Generous Maximum ( M_2 )</td>
<td>(1, 1, 3, 0, 0)</td>
<td>12</td>
</tr>
<tr>
<td>Minimum Cost Maximum ( M_3 )</td>
<td>(1, 3, 0, 1, 0)</td>
<td>11</td>
</tr>
</tbody>
</table>
Profile-based and Cost-based optimality criterion

students' preferences

\[ s_1 : p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \]
\[ s_2 : p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \]
\[ s_3 : p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \]
\[ s_4 : p_1 \quad p_3 \quad p_5 \quad p_4 \quad p_2 \]
\[ s_5 : p_2 \quad p_5 \quad p_4 \quad p_3 \quad p_1 \]

\[ M_1 = \{(s_1, p_1), (s_2, p_4), (s_3, p_5), (s_4, p_3), (s_5, p_2)\} \]
\[ M_2 = \{(s_1, p_1), (s_2, p_2), (s_3, p_3), (s_4, p_5), (s_5, p_4)\} \]
\[ M_3 = \{(s_1, p_1), (s_2, p_2), (s_3, p_4), (s_4, p_3), (s_5, p_5)\} \]

<table>
<thead>
<tr>
<th>Matching ( M_i )</th>
<th>Profile ( \rho(M_i) )</th>
<th>Cost ( \lambda(M_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy Maximum ( M_1 )</td>
<td>(2, 1, 0, 1, 1)</td>
<td>13</td>
</tr>
<tr>
<td>Generous Maximum ( M_2 )</td>
<td>(1, 1, 3, 0, 0)</td>
<td>12</td>
</tr>
<tr>
<td>Minimum Cost Maximum ( M_3 )</td>
<td>(1, 3, 0, 1, 0)</td>
<td>11</td>
</tr>
</tbody>
</table>
Profile-based and Cost-based optimality criterion

students' preferences

\[
\begin{align*}
s_1 : p_1 & \quad p_2 & \quad p_3 & \quad p_4 & \quad p_5 \\
s_2 : p_1 & \quad p_2 & \quad p_3 & \quad p_4 & \quad p_5 \\
s_3 : p_1 & \quad p_2 & \quad p_3 & \quad p_4 & \quad p_5 \\
s_4 : p_1 & \quad p_3 & \quad p_5 & \quad p_4 & \quad p_2 \\
s_5 : p_2 & \quad p_5 & \quad p_4 & \quad p_3 & \quad p_1
\end{align*}
\]

\[
M_1 = \{(s_1, p_1), (s_2, p_4), (s_3, p_5), (s_4, p_3), (s_5, p_2)\}
\]

\[
M_2 = \{(s_1, p_1), (s_2, p_2), (s_3, p_3), (s_4, p_5), (s_5, p_4)\}
\]

\[
M_3 = \{(s_1, p_1), (s_2, p_2), (s_3, p_4), (s_4, p_3), (s_5, p_5)\}
\]

<table>
<thead>
<tr>
<th>Matching $M_i$</th>
<th>Profile $\rho(M_i)$</th>
<th>Cost $\lambda(M_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy Maximum $M_1$</td>
<td>$(2, 1, 0, 1, 1)$</td>
<td>13</td>
</tr>
<tr>
<td>Generous Maximum $M_2$</td>
<td>$(1, 1, 3, 0, 0)$</td>
<td>12</td>
</tr>
<tr>
<td>Minimum Cost Maximum $M_3$</td>
<td>$(1, 3, 0, 1, 0)$</td>
<td>11</td>
</tr>
</tbody>
</table>
Adapting Capacitated House Allocation (CHA) algorithms

- CHA is a many-to-one matching problem with preferences on one side only.
  - SPA is a generalisation of CHA.

- However not all SPA requirements exist in the CHA context.
  - Lecturer upper and lower quotas.
  - Project lower quotas.

- In the absence of these requirements CHA algorithms can be used to solve SPA problems.
  - $O(Rm\sqrt{n} \log(n))$ by Mehlhorn et al '06.
  - $O(Rmn^2)$ by Sng '08.
  - $O(Rm\sqrt{n})$ by Huang et al '13.
Adapting Capacitated House Allocation (CHA) algorithms

- CHA is a many-to-one matching problem with preferences on one side only.
  - SPA is a generalisation of CHA.

- However not all SPA requirements exist in the CHA context.
  - Lecturer upper and lower quotas.
  - Project lower quotas.

- In the absence of these requirements CHA algorithms can be used to solve SPA problems.
  - $O(Rm\sqrt{n} \log(n))$ by Mehlhorn et al '06.
  - $O(Rmn^2)$ by Sng '08.
  - $O(Rm\sqrt{n})$ by Huang et al '13.
Adapting Capacitated House Allocation (CHA) algorithms

- CHA is a many-to-one matching problem with preferences on one side only.
  - SPA is a generalisation of CHA.

- However not all SPA requirements exist in the CHA context.
  - Lecturer upper and lower quotas.
  - Project lower quotas.

- In the absence of these requirements CHA algorithms can be used to solve SPA problems.
  - $O(Rm\sqrt{n} \log(n))$ by Mehlhorn et al '06.
  - $O(Rmn^2)$ by Sng '08.
  - $O(Rm\sqrt{n})$ by Huang et al '13.
A network flow approach

- Abraham '07 and Zelvyte '14 proposed network flow models for SPA.

- This allows for potentially greater flexibility.

- Both models find a minimum cost maximum matching efficiently.

- By correctly assigning edge weights, Zelvyte's model also finds all profile-based optimal matchings.
  - However the exponentially large edge weights may render this approach infeasible for large instances.

- We extend Sng and Irving's CHA algorithm to the network flow context thus finding these profile-based optimal SPA matchings in \( O(Rn_1^2(m_2 + n_2^2)) \) time.
A network flow approach

- Abraham '07 and Zelvyte '14 proposed network flow models for SPA.
- This allows for potentially greater flexibility.
- Both models find a minimum cost maximum matching efficiently.
- By correctly assigning edge weights, Zelvyte's model also finds all profile-based optimal matchings.
  - However the exponentially large edge weights may render this approach infeasible for large instances.
- We extend Sng and Irving's CHA algorithm to the network flow context thus finding these profile-based optimal SPA matchings in $O(Rn_1^2(m_2 + n_2^2))$ time.
A network flow approach

- Abraham '07 and Zelvyte '14 proposed network flow models for SPA.
- This allows for potentially greater flexibility.
- Both models find a minimum cost maximum matching efficiently.
- By correctly assigning edge weights, Zelvyte's model also finds all profile-based optimal matchings.
  - However the exponentially large edge weights may render this approach infeasible for large instances.
- We extend Sng and Irving's CHA algorithm to the network flow context thus finding these profile-based optimal SPA matchings in $O(Rn_1^2(m_2 + n_2^2))$ time.
A network flow approach

- Abraham ’07 and Zelvyte ’14 proposed network flow models for SPA.
- This allows for potentially greater flexibility.
- Both models find a minimum cost maximum matching efficiently.
- By correctly assigning edge weights, Zelvyte's model also finds all profile-based optimal matchings.
  - However the exponentially large edge weights may render this approach infeasible for large instances.

- We extend Sng and Irving's CHA algorithm to the network flow context thus finding these profile-based optimal SPA matchings in $O(Rn_1^2(m_2 + n_2^2))$ time.
A network flow approach

- Abraham '07 and Zelvyte '14 proposed network flow models for SPA.
- This allows for potentially greater flexibility.
- Both models find a minimum cost maximum matching efficiently.
- By correctly assigning edge weights, Zelvyte's model also finds all profile-based optimal matchings.
  - However the exponentially large edge weights may render this approach infeasible for large instances.
- We extend Sng and Irving's CHA algorithm to the network flow context thus finding these profile-based optimal SPA matchings in $O(Rn_1^2(m_2 + n_2^2))$ time.
1. Motivation
1.1 The Student/Project Allocation problem (SPA)
1.2 Profile-based and Cost-based optimality criterion

2. Greedy maximum matchings in SPA
2.1 A network flow model
2.2 Algorithm GREEDY-MAX-SPA

3. Other extensions
3.1 Lecturer lower quotas
3.2 Project lower quotas
A network flow model

students' preferences:
\[ s_1 : p_1 \quad p_3 \]
\[ s_2 : p_2 \quad p_1 \]
\[ s_3 : p_2 \]

lecturers' offerings:
\[ l_1 : \{ p_1 \} \]
\[ l_2 : \{ p_2, p_3 \} \]

\[ c_1 = 2, \quad c_2 = c_3 = 1, \quad d_1 = 2, \quad d_2 = 1 \]
A network flow model

students' preferences:
\[ s_1 : p_1 \quad p_3 \]
\[ s_2 : p_2 \quad p_1 \]
\[ s_3 : p_2 \]

lecturers' offerings:
\[ l_1 : \{ p_1 \} \]
\[ l_2 : \{ p_2, p_3 \} \]

\[ c_1 = 2, \ c_2 = c_3 = 1, \ d_1 = 2, \ d_2 = 1 \]
A network flow model

students' preferences:

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

lecturers' offerings:

\[ l_1 : \{p_1\} \]
\[ l_2 : \{p_2, p_3\} \]

\[ c_1 = 2, \quad c_2 = c_3 = 1, \quad d_1 = 2, \quad d_2 = 1 \]
A network flow model

students' preferences:

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

lecturers' offerings:

\[ l_1 : \{ p_1 \} \]
\[ l_2 : \{ p_2, p_3 \} \]

\[ c_1 = 2, \quad c_2 = c_3 = 1, \quad d_1 = 2, \quad d_2 = 1 \]
A network flow model

students' preferences:
- $s_1 : p_1 \ p_3$
- $s_2 : p_2 \ p_1$
- $s_3 : p_2$

lecturers' offerings:
- $l_1 : \{p_1\}$
- $l_2 : \{p_2, p_3\}$

$c_1 = 2, \ c_2 = c_3 = 1, \ d_1 = 2, \ d_2 = 1$
A network flow model

students' preferences:
\[ s_1 : p_1 \quad p_3 \]
\[ s_2 : p_2 \quad p_1 \]
\[ s_3 : p_2 \]

lecturers' offerings:
\[ l_1 : \{ p_1 \} \]
\[ l_2 : \{ p_2, p_3 \} \]

\[ c_1 = 2, \quad c_2 = c_3 = 1, \quad d_1 = 2, \quad d_2 = 1 \]
A network flow model

students' preferences:
\[ s_1 : p_1 \quad p_3 \]
\[ s_2 : p_2 \quad p_1 \]
\[ s_3 : p_2 \]

lecturers' offerings:
\[ l_1 : \{p_1\} \]
\[ l_2 : \{p_2, p_3\} \]

\[ c_1 = 2, \quad c_2 = c_3 = 1, \quad d_1 = 2, \quad d_2 = 1 \]
A network flow model

students' preferences:

$s_1 : p_1 \ p_3$
$s_2 : p_2 \ p_1$
$s_3 : p_2$

lecturers' offerings:

$l_1 : \{p_1\}$
$l_2 : \{p_2, p_3\}$

c_1 = 2, c_2 = c_3 = 1, d_1 = 2, d_2 = 1
A network flow model

students' preferences:

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

lecturers' offerings:

\[ l_1 : \{p_1\} \]
\[ l_2 : \{p_2, p_3\} \]

\[ c_1 = 2, \quad c_2 = c_3 = 1, \quad d_1 = 2, \quad d_2 = 1 \]
A network flow model

students' preferences:
\[ s_1 : p_1 \, p_3 \]
\[ s_2 : p_2 \, p_1 \]
\[ s_3 : p_2 \]

lecturers' offerings:
\[ l_1 : \{ p_1 \} \]
\[ l_2 : \{ p_2, p_3 \} \]

\[ c_1 = 2, \; c_2 = c_3 = 1, \; d_1 = 2, \; d_2 = 1 \]
1. Motivation
1.1 The Student/Project Allocation problem (SPA)
1.2 Profile-based and Cost-based optimality criterion

2. Greedy maximum matchings in SPA
2.1 A network flow model
2.2 Algorithm GREEDY-MAX-SPA

3. Other extensions
3.1 Lecturer lower quotas
3.2 Project lower quotas
Algorithm GREEDY-MAX-SPA

**Require:** SPA instance $I$;

**Ensure:** return matching $M$;

1: define flow network $N(I) = \langle G, c \rangle$;
2: define empty flow $f$;
3: loop
4: $P =$ maximum profile augmenting path in $N(I)$ w.r.t. $f$;
5: if $P \neq \text{null}$ then
6: augment $f$ along $P$;
7: else
8: return the corresponding matching $M(f)$;
9: end if
10: end loop
Algorithm GREEDY-MAX-SPA

Require: SPA instance $I$;
Ensure: return matching $M$;
1: define flow network $N(I) = \langle G, c \rangle$;
2: define empty flow $f$;
3: loop
4: $P =$ maximum profile augmenting path in $N(I)$ w.r.t. $f$;
5: if $P \neq \text{null}$ then
6: augment $f$ along $P$;
7: else
8: return the corresponding matching $M(f)$;
9: end if
10: end loop
Algorithm GREEDY-MAX-SPA

Require: SPA instance $I$;  
Ensure: return matching $M$;  
1: define flow network $N(I) = \langle G, c \rangle$;  
2: define empty flow $f$;  
3: loop  
4: $P =$ maximum profile augmenting path in $N(I)$ w.r.t. $f$;  
5: if $P \neq \text{null}$ then  
6: augment $f$ along $P$;  
7: else  
8: return the corresponding matching $M(f)$;  
9: end if  
10: end loop
Algorithm GREEDY-MAX-SPA

Require: SPA instance $I$;
Ensure: return matching $M$;
1: define flow network $N(I) = (G, c)$;
2: define empty flow $f$;
3: loop
4: $P =$ maximum profile augmenting path in $N(I)$ w.r.t. $f$;
5: if $P \neq null$ then
6: augment $f$ along $P$;
7: else
8: return the corresponding matching $M(f)$;
9: end if
10: end loop
Algorithm GREEDY-MAX-SPA

**Require:** SPA instance $I$;

**Ensure:** return matching $M$;

1. define flow network $N(I) = \langle G, c \rangle$;
2. define empty flow $f$;

3. **loop**
4. $P =$ maximum profile augmenting path in $N(I)$ w.r.t. $f$;
5. **if** $P \neq \text{null}$ **then**
6. augment $f$ along $P$;
7. **else**
8. return the corresponding matching $M(f)$;
9. **end if**
10. **end loop**
Algorithm GREEDY-MAX-SPA

Require: SPA instance $I$;
Ensure: return matching $M$;

1: define flow network $N(I) = \langle G, c \rangle$;
2: define empty flow $f$;

3: loop
4: $P =$ maximum profile augmenting path in $N(I)$ w.r.t. $f$;
5: if $P \neq$ null then
6: augment $f$ along $P$;
7: else
8: return the corresponding matching $M(f)$;
9: end if
10: end loop
Algorithm **GREEDY-MAX-SPA**

- Based on the Ford-Fulkerson algorithm for finding a maximum flow in a network.

- A maximum profile augmenting path in $N(I)$ w.r.t. $f$:
  - augments $f$ by 1.
  - provides the best improvement to the profile of $M(f)$ w.r.t. greedy criteria.
  - is found using ideas from the Bellman-Ford algorithm for the single source shortest path problem.

**Lemma**

$M(f)$ is a greedy $k$-matching at every stage within the main loop of **GREEDY-MAX-SPA** where $k = |M(f)|$.

- For generous maximum matchings we seek to find a minimum profile augmenting path in $N(I)$ w.r.t. $f$.
Algorithm GREEDY-MAX-SPA

- Based on the Ford-Fulkerson algorithm for finding a maximum flow in a network.

- A maximum profile augmenting path in $N(I)$ w.r.t. $f$:
  - augments $f$ by 1.
  - provides the best improvement to the profile of $M(f)$ w.r.t. greedy criteria.
  - is found using ideas from the Bellman-Ford algorithm for the single source shortest path problem.

**Lemma**

$M(f)$ is a greedy $k$-matching at every stage within the main loop of GREEDY-MAX-SPA where $k = |M(f)|$.

- For generous maximum matchings we seek to find a minimum profile augmenting path in $N(I)$ w.r.t. $f$. 
Algorithm GREEDY-MAX-SPA

- Based on the Ford-Fulkerson algorithm for finding a maximum flow in a network.

- A **maximum profile augmenting path** in $N(I)$ w.r.t. $f$:
  - augments $f$ by 1.
  - provides the best improvement to the profile of $M(f)$ w.r.t. greedy criteria.
  - is found using ideas from the Bellman-Ford algorithm for the single source shortest path problem.

**Lemma**

$M(f)$ is a greedy $k$-matching at every stage within the main loop of GREEDY-MAX-SPA where $k = |M(f)|$.

- For generous maximum matchings we seek to find a minimum profile augmenting path in $N(I)$ w.r.t. $f$. 

Based on the Ford-Fulkerson algorithm for finding a maximum flow in a network.

A maximum profile augmenting path in $N(I)$ w.r.t. $f$:

- augments $f$ by 1.
- provides the best improvement to the profile of $M(f)$ w.r.t. greedy criteria.
- is found using ideas from the Bellman-Ford algorithm for the single source shortest path problem.

**Lemma**

$M(f)$ is a greedy $k$-matching at every stage within the main loop of GREEDY-MAX-SPA where $k = |M(f)|$.

For generous maximum matchings we seek to find a minimum profile augmenting path in $N(I)$ w.r.t. $f$. 
Finding a maximum profile augmenting path

- We can switch projects assigned to a student in order to extend the augmenting path.
  - Switch \((s_1, p_2)\) for \((s_1, p_1)\).

\[
\rho(M) = \{0, 1\}
\]
Finding a maximum profile augmenting path

- We can switch projects assigned to a student in order to extend the augmenting path.
  - Switch \((s_1, p_2)\) for \((s_1, p_1)\).

\[
\rho(M) = \{0, 1\}
\]
Finding a maximum profile augmenting path

- We can switch projects assigned to a student in order to extend the augmenting path.
  - Switch \((s_1, p_2)\) for \((s_1, p_1)\).

\[
\rho(M) = \{2, 0\}
\]
We can switch student assignments between projects offered by the same lecturer in order to extend the augmenting path.

Switch \((p_3, l_2)\) for \((p_2, l_2)\).

\[
\begin{align*}
\rho(\mathcal{M}) &= \{0, 1\}
\end{align*}
\]
Finding a maximum profile augmenting path

- We can switch student assignments between projects offered by the same lecturer in order to extend the augmenting path.
  - Switch \((p_3, l_2)\) for \((p_2, l_2)\).

\[
\begin{align*}
\rho(M) &= \{0, 1\}
\end{align*}
\]
Finding a maximum profile augmenting path

- We can switch student assignments between projects offered by the same lecturer in order to extend the augmenting path.
  - Switch \((p_3, l_2)\) for \((p_2, l_2)\).

\[ \rho(M) = \{2, 0\} \]
1. Motivation
1.1 The Student/Project Allocation problem (SPA)
1.2 Profile-based and Cost-based optimality criterion

2. Greedy maximum matchings in SPA
2.1 A network flow model
2.2 Algorithm GREEDY-MAX-SPA

3. Other extensions
3.1 Lecturer lower quotas
3.2 Project lower quotas
Lecturer lower quotas (SPA-LL)

**Require:** SPA-LL instance $I$;

**Ensure:** return matching $M$;

1: /*construct an instance $I'$ of SPA as follows*/
2: $I' = I$;

3: for each lecturer $l_k$ do
4: set lower quota of $l_k$ in $I'$ to 0;
5: set upper quota of $l_k$ in $I'$ to lower quota of $l_k$ in $I$;
6: end for

7: $M' = \text{GREEDY-MAX-SPA}(I')$;

8: if corresponding flow $f(M')$ saturates all lecturers in $I'$ then
9: set upper quota of $l_k$ in $I'$ to upper quota of $l_k$ in $I$;
10: $M = \text{GREEDY-MAX-SPA}(I')$; /* setting initial flow to $f(M')$*/
11: else
12: return as unsolvable;
13: end if
Lecturer lower quotas (SPA-LL)

**Require:** SPA-LL instance $I$;

**Ensure:** return matching $M$;

1: /*construct an instance $I'$ of SPA as follows*/
2: $I' = I$;

3: for each lecturer $l_k$ do
4:   set lower quota of $l_k$ in $I'$ to 0;
5:   set upper quota of $l_k$ in $I'$ to lower quota of $l_k$ in $I$;
6: end for

7: $M' = \text{GREEDY-MAX-SPA}(I')$;

8: if corresponding flow $f(M')$ saturates all lecturers in $I'$ then
9:   set upper quota of $l_k$ in $I'$ to upper quota of $l_k$ in $I$;
10: $M = \text{GREEDY-MAX-SPA}(I')$; /* setting initial flow to $f(M')$*/
11: else
12:   return as unsolvable;
13: end if
**Lecturer lower quotas (SPA-LL)**

**Require:** SPA-LL instance $I$;

**Ensure:** return matching $M$;

1. /*construct an instance $I'$ of SPA as follows*/
2. $I' = I$;

3. **for** each lecturer $l_k$ **do**
4. set lower quota of $l_k$ in $I'$ to 0;
5. set upper quota of $l_k$ in $I'$ to lower quota of $l_k$ in $I$;
6. **end for**

7. $M' = \text{GREEDY-MAX-SPA}(I')$;

8. **if** corresponding flow $f(M')$ saturates all lecturers in $I'$ **then**
9. set upper quota of $l_k$ in $I'$ to upper quota of $l_k$ in $I$;
10. $M = \text{GREEDY-MAX-SPA}(I')$; /* setting initial flow to $f(M')$*/
11. **else**
12. return as unsolvable;
13. **end if**
Lecturer lower quotas (SPA-LL)

**Require:** SPA-LL instance $I$;

**Ensure:** return matching $M$;

1: /*construct an instance $I'$ of SPA as follows*/
2: $I' = I$;

3: for each lecturer $l_k$ do
4: set lower quota of $l_k$ in $I'$ to 0;
5: set upper quota of $l_k$ in $I'$ to lower quota of $l_k$ in $I$;
6: end for

7: $M' = \text{GREEDY-MAX-SPA}(I')$;

8: if corresponding flow $f(M')$ saturates all lecturers in $I'$ then
9: set upper quota of $l_k$ in $I'$ to upper quota of $l_k$ in $I$;
10: $M = \text{GREEDY-MAX-SPA}(I')$; /* setting initial flow to $f(M')$*/
11: else
12: return as unsolvable;
13: end if
Lecturer lower quotas (SPA-LL)

Require: SPA-LL instance $I$;
Ensure: return matching $M$;

1: /*construct an instance $I'$ of SPA as follows*/
2: $I' = I$;
3: for each lecturer $l_k$ do
4: set lower quota of $l_k$ in $I'$ to 0;
5: set upper quota of $l_k$ in $I'$ to lower quota of $l_k$ in $I$;
6: end for
7: $M' = \text{GREEDY-MAX-SPA}(I')$;
8: if corresponding flow $f(M')$ saturates all lecturers in $I'$ then
9: set upper quota of $l_k$ in $I'$ to upper quota of $l_k$ in $I$;
10: $M = \text{GREEDY-MAX-SPA}(I')$; /* setting initial flow to $f(M')$*/
11: else
12: return as unsolvable;
13: end if

Kwanashie et al. Profile-based optimal matchings in SPA BCTCS, April 2014 23 / 26
Lecturer lower quotas (SPA-LL)

**Require:** SPA-LL instance $I$;

**Ensure:** return matching $M$;

1: /*construct an instance $I'$ of SPA as follows*/
2: $I' = I$;

3: for each lecturer $l_k$ do
4: set lower quota of $l_k$ in $I'$ to 0;
5: set upper quota of $l_k$ in $I'$ to lower quota of $l_k$ in $I$;
6: end for

7: $M' = \text{GREEDY-MAX-SPA}(I')$;

8: if corresponding flow $f(M')$ saturates all lecturers in $I'$ then
9: set upper quota of $l_k$ in $I'$ to upper quota of $l_k$ in $I$;
10: $M = \text{GREEDY-MAX-SPA}(I')$; /* setting initial flow to $f(M')$*/
11: else
12: return as unsolvable;
13: end if
Outline

1. Motivation
   1.1 The Student/Project Allocation problem (SPA)
   1.2 Profile-based and Cost-based optimality criterion

2. Greedy maximum matchings in SPA
   2.1 A network flow model
   2.2 Algorithm GREEDY-MAX-SPA

3. Other extensions
   3.1 Lecturer lower quotas
   3.2 Project lower quotas
• If no matching exists in which all projects meet their lower quotas.
  • We may report the problem as unsolvable.
  • We may remove the project from consideration.

• For the former scenario, we are looking to extend our algorithm to solve it.

• The latter scenario has been shown to be NP-hard.
  • Can we find a good approximation algorithms?
If no matching exists in which all projects meet their lower quotas.
  - We may report the problem as unsolvable.
  - We may remove the project from consideration.

For the former scenario, we are looking to extend our algorithm to solve it.

The latter scenario has been shown to be NP-hard.
  - Can we find a good approximation algorithms?
Project lower quotas (SPA-PL)

- If no matching exists in which all projects meet their lower quotas.
  - We may report the problem as unsolvable.
  - We may remove the project from consideration.

- For the former scenario, we are looking to extend our algorithm to solve it.

- The latter scenario has been shown to be NP-hard.
  - Can we find a good approximation algorithms?
Thank You.