On Stable Marriages and Greedy Matchings

Fredrik Manne
University of Bergen, Norway

Md. Naim, Håkon Lerring, Mahantesh Halappanavar
Background

The Stable Marriage (SM) problem has a long and rigorous history.

Greedy Matchings (GM) have applications in CSC applications.

Objective:
- Formalize the connections between the Stable Marriage problem and computing Greedy Matchings.

Consequences:
- Many "new" algorithms for computing GM are variants of algorithms for SM.
- Parallel algorithms for computing GM can also be applied to SM.
The Stable Marriage Problem

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>((w_3, w_2, w_1, w_4))</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>((w_1, w_4, w_3, w_2))</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>((w_1, w_2, w_3, w_4))</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>((w_2, w_4, w_3, w_1))</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>((m_4, m_2, m_3, m_1))</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>((m_1, m_2, m_3, m_4))</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>((m_3, m_2, m_1, m_4))</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>((m_2, m_4, m_3, m_1))</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Gale-Shapley Algorithm

1 2 3 4
(w3, w2, w1, w4)

1 2 3 4
(m4, m2, m3, m1)

(w1, w4, w3, w2)

(m1, m2, m3, m4)

(w1, w2, w3, w4)

(m3, m2, m1, m4)

(w2, w4, w3, w1)

(m2, m4, m3, m1)
The McVitie-Wilson Algorithm

1 2 3 4
(w₃, w₂, w₁, w₄)

(m₄, m₂, m₃, m₁)

1 2 3 4
(w₁, w₄, w₃, w₂)

(m₁, m₂, m₃, m₄)

1 2 3 4
(w₁, w₂, w₃, w₄)

(m₃, m₂, m₁, m₄)

1 2 3 4
(w₂, w₄, w₃, w₁)

(m₂, m₄, m₃, m₁)

Implementation: GS uses a queue while MW uses a stack for the remaining men
Greedy Matching

M = Ø
While there are edges remaining
    e = heaviest edge
    M = M ∪ {e}
    remove edges incident on e

\[ w(\text{greedy}) \geq 0.5 \, w(\text{optimal}) \]
Computing a Greedy Matching using a Stable Marriage algorithm

![Diagram showing a matching process in a stable marriage algorithm with two sets of candidates and their preferences.](image-url)
Using McVitie-Wilson directly

This is exactly the suitor-algorithm [Manne & Halappanavar 14]

which builds on:
- The pointer algorithm [Manne & Bisseling 08]
- The Preis algorithm [Preis 99]
Going parallel

Threads run either the Gayle-Shapley or the McVitie-Wilson algorithm

Use compare-and-swap to protect "women"

Implementations using both OpenMP and GPU
“Easy” problems

- Each man selects between $\log n$ and $2\log n$ women and ranks randomly.
- Women rank men who rank them (randomly).
- Expected total work $n \log n$

OpenMP: Using 36 threads on two Intel Xeon E5-2699 processors
GPU: Tesla K40m with 2880 cores
“Hard” problems

- Each man uses the same total ranking of all the women
- Each woman uses the same total ranking of all the men.

- Expect high contention for the same “women”
- Total work will be \((n+1)n/2\)
Concluding Remarks

• Recent greedy b-matching algorithms also follow directly from algorithms for the many-to-many stable assignment problem.

• Major open question [Manlove 13]: Is Stable Marriage in NC?
  ‣ Maybe not so relevant…

• Stable Marriage assumes sorted priority lists, whereas Greedy Matching makes no such assumption.
  ‣ Preis solved Greedy Matching in O(m) time.
  ‣ Can Stable Matching with unsorted weighted priority lists also be solved in O(m) time?
Gale-Shapley implementation

Place all vertices in queue $Q$
while $Q \neq \emptyset$
    $u = Q.\text{first}()$
    $p = \text{nextCandidate}(u)$
    while $r_p(u) > r_p(\text{suitor}(p))$
        $p = \text{nextCandidate}(u)$
    if $\text{suitor}(p) \neq \text{null}$
        $Q.\text{add}(\text{suitor}(p))$
    $\text{suitor}(p) = u$