Some applications of the Matroid Theorem

**Knapsack**

\[ B = 10 \]

\[ x_1, x_2, x_3 \]

Weight: 6 3 9

Maximal feasible subsets have different cardinalities:

\[ \{ x_1, x_2 \} \quad \{ x_3 \} \]

**Matching**

Maximal subset:

\[ \{(a,c), (b,d)\} \quad \{(b,c)\} \]

Greedy doesn't work for Knapsack matching
Job scheduling problem

Set of jobs $J_1, J_2, \ldots, J_n$

Time spans $\Delta_1, \Delta_2, \ldots, \Delta_n$

Deadlines $d_1, d_2, \ldots, d_n$

Penalty $p_1, p_2, \ldots, p_n$

Goal: Schedule all jobs so as to minimize penalty incurred

Special case $\Delta_i = 1$

Example

$J_1, J_2, J_3$

deadlines

penalty

1 2 1

5 2 8

A schedule is a mapping of jobs to time slots, such that no more than one job can be done in the same time unit.
Possible schedules \( \{ J_1, J_2 \} \) (reverse cannot be scheduled)

Penalty 8

\[ \checkmark \] \( \{ J_3, J_2 \} \)

Any subset of a feasible schedule is also feasible.

Subset system framework:

\[ S : \text{set of jobs} \]

Independent subset \( J : \text{subset of jobs that can be scheduled without missing deadlines} \)

Objective: Minimizing the penalty of the jobs not scheduled is equivalent to maximizing penalty of the scheduled

Note: The depend of independent subsets do not have the scheduling information

Problem: Given a set of feasible jobs, how do we actually schedule
schedule: $J_{i_1}, J_{i_2}, J_{i_3}, J_{i_4}, \ldots, J_{i_k}$

Time slots: $t_1, t_2, t_3, \ldots, t_k$

Deadlines: $d_1, d_2, d_3, \ldots, d_k$

If it is feasible then $t_i \leq d_i$

Suppose $J_a$ is scheduled before $J_b$

$t_a < t_b$

but $d_a > d_b$

Claim: Swapping the time slots $J_a J_b$ will preserve feasibility.

Problem: Given a set of feasible jobs, design an efficient algorithm for scheduling.

Observation: Time slots:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{50}$</td>
<td>$J_5$</td>
<td>$J_3$</td>
<td>$J_20$</td>
<td>$J_{i_6}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


In a feasible schedule with gaps, we can maintain feasibility by removing the gaps.

**Is it a Matroid?**

We will try to prove the exchange property.

We have two sets of feasible jobs, say \( S_1 \) and \( S_2 \) such that \( |S_2| = |S_1| + 1 \).

Can we find a job \( J' \in S_2 - S_1 \) such that \( S_1 \cup \{J'\} \) is feasible?

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & \ldots & i & k & k+1 \\
S_1 & J_1 & J_2 & J_3 & & J_i & & \\
S_2 & J'_1 & J'_2 & J'_3 & & J'_i & & J'_{k+1} \\
\end{array}
\]

We have feasible schedule of \( S_1 \) and \( S_2 \).

**Case 1:** \( J'_{k+1} \notin S_1 \). Easy: Include \( J'_{k+1} \) in \( S_1 \).
Case 2 \( J_{k+1} \in S_l \). Suppose \( J_i = J_{k+1} \).

Then we can transform the schedule:

<table>
<thead>
<tr>
<th>( S_1 )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>\ldots</th>
<th>( K )</th>
<th>( K+1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_2 )</td>
<td></td>
<td></td>
<td></td>
<td>( J_i )</td>
<td>( J_i )</td>
<td>( J_i )</td>
</tr>
</tbody>
</table>

Apply the same argument to \( S_1 - J_i \) and \( S_2 - J_i \)

until \( S_1 \) has no jobs and \( S_2 \) has one job. Clearly we can include that job to \( S_1 \).

Exchange property holds and then for greedy works.

Some points of a Generic greedy

1. Even if generic greedy doesn't work (i.e., not matroid) some other version of greedy may work (e.g., Prim's).

2. Even if greedy doesn't work to give the maximum profit, it may still be effective.
Effective: May still give some guarantees like 50% of the maximum et.

**Knapsack problem**

- \( B = 13 \)
- \( x_1, x_2, x_3 \)
- \( w_b = 5, 9, 8 \)
- \( \text{profit} = 20, 40, 30 \)
- \( \text{ratio} = 4 \cdot 4 < 3 \)

Suppose we look at profit/wb ratio.

Choose the object with the best ratio until we cannot add any more. Suppose the decreasing order of ratios is \( y_1, y_2, y_3, \ldots, y_k, y_{k+1} \) where all are full.

Choose \( \max \{ s y_1, y_2, \ldots, y_3, y_{k+1} \} \)
This guarantees a solution that is at least $\frac{1}{2}$ as good as optimum.