Software Project Scheduling Problem

Leandro Minku
www.cs.bham.ac.uk/~minkull

Nature-inspired Optimisation Lecture
Outline

• What is the Software Project Scheduling Problem (SPSP)?
• Why are automated optimisation methods important for the SPSP?
• How to solve the SPSP?
Software Project Scheduling Problem (SPSP)

Each task requires a set of skills and effort. Tasks also have a precedence relation.

Different allocations will lead to different project cost and duration.
Software Project Scheduling Problem (SPSP)

SPSP: find a good allocation of employees to tasks in a software project so as to minimise its cost and completion time.

It is very difficult to optimally assign employees to tasks manually.

• The space of possible allocations can be enormous.

We can use optimisation algorithms (e.g., EAs) to solve the SPSP!
Advantages of Optimisation Algorithms for the SPSP

- Insight into how to optimise objectives -- they may find solutions that no human has thought of.
- Speed up the task of allocating employees to tasks.
- Help software manager to find solutions that satisfy all constraints.
  - Team must have skills to perform a task.
  - No overwork is allowed [video].
Setting: assume we are given
• $n$ employees $e_1, \ldots, e_n$ with salaries and sets of skills;
• $m$ tasks $t_1, \ldots, t_m$ with efforts and sets of required skills;
• a task precedence graph (TPG).

Problem: allocate employees to tasks so as to:
• **minimise cost** (total salaries paid) and
• **minimise duration** (completion time).

Constraints:
• **team must have required skills and**
• **no overwork.**
Solving the SPSP

What type of evolutionary algorithm would be adequate?

- Multi-objective Evolutionary Algorithms, e.g., NSGA-II.

\[ f_1 = \text{cost} \]
\[ f_2 = \text{duration} \]

Non-dominated solutions.

Solution \( x_1 \) dominates \( x_2 \) iif:

\[ \forall i, f_i(x_1) \leq f_i(x_2) \]
\[ \exists i, f_i(x_1) < f_i(x_2) \]
NSGA-II

- Step 1: For a population of size $M$, create a group of $M$ offspring using the desired crossover and mutation operators
- Step 2: The offspring and their parent solutions are combined into a group of size $2M$
- Step 3: Selecting the fittest $M$ individuals from this group as parents for the next generation by:
  - Step 3.1: Nondominated Sorting: similar to NSGA, e.g., to identify all non-dominated fronts and sort them
  - Step 3.2: **Crowding distance sorting**: removes the “most crowded” individuals, e.g., those individual with small $D_i$ from this final front, in order to make it fit into the group of $M$ parents.

* From Lecture 15.

[Video: https://youtu.be/sEEiGM9em8s]
Designing an Evolutionary Algorithm

- Representation / encoding;
- mutation and crossover;
- fitness / objectives evaluation;
- how to deal with constraints.
Designing an Evolutionary Algorithm

- Representation / encoding;
- mutation and crossover;
- fitness / objectives evaluation;
- how to deal with constraints.
Formulating the SPSP

Setting: assume we are given
- \( n \) employees \( e_1, \ldots, e_n \) with salaries and sets of skills;
- \( m \) tasks \( t_1, \ldots, t_m \) with efforts and sets of required skills;
- a task precedence graph (TPG).

Problem: allocate employees to tasks so as to:
- minimise cost (total salaries paid) and
- minimise duration (completion time).

Constraints:
- team must have required skills and
- no overwork.
## Representation

**Dedication:** percentage of time an employee spends on a task, respecting a certain granularity $k$.

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>...</th>
<th>$t_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>$x_{1,1}$</td>
<td>$x_{1,2}$</td>
<td>...</td>
<td>$x_{1,m}$</td>
</tr>
<tr>
<td>$e_2$</td>
<td>$x_{2,1}$</td>
<td>$x_{2,2}$</td>
<td>...</td>
<td>$x_{2,m}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$e_n$</td>
<td>$x_{n,1}$</td>
<td>$x_{n,2}$</td>
<td>...</td>
<td>$x_{n,m}$</td>
</tr>
</tbody>
</table>

$x_{i,j} \in \left\{ \frac{0}{k}, \frac{1}{k}, \frac{2}{k}, ..., \frac{k}{k} \right\}$

Employees can divide their attention among tasks.
Designing an Evolutionary Algorithm

• Representation / encoding;
• mutation and crossover;
• fitness / objectives evaluation;
• how to deal with constraints.
Mutation and Crossover

- Mutation of $x_{i,j}$ picks a new dedication uniformly at random from $\{0/k, 1/k, \ldots, k/k\} \setminus x_{i,j}$.

- Crossover: exchange rows

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>$X_{1,1}$</td>
<td>$X_{1,2}$</td>
<td>$X_{1,3}$</td>
<td>$X_{1,4}$</td>
</tr>
<tr>
<td>$e_2$</td>
<td>$X_{2,1}$</td>
<td>$X_{2,2}$</td>
<td>$X_{2,3}$</td>
<td>$X_{2,4}$</td>
</tr>
<tr>
<td>$e_3$</td>
<td>$X_{3,1}$</td>
<td>$X_{3,2}$</td>
<td>$X_{3,3}$</td>
<td>$X_{3,4}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>$X_{1,1}$</td>
<td>$X_{1,2}$</td>
<td>$X_{1,3}$</td>
<td>$X_{1,4}$</td>
</tr>
<tr>
<td>$e_2$</td>
<td>$X_{2,1}$</td>
<td>$X_{2,2}$</td>
<td>$X_{2,3}$</td>
<td>$X_{2,4}$</td>
</tr>
<tr>
<td>$e_3$</td>
<td>$X_{3,1}$</td>
<td>$X_{3,2}$</td>
<td>$X_{3,3}$</td>
<td>$X_{3,4}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>$X_{1,1}$</td>
<td>$X_{1,2}$</td>
<td>$X_{1,3}$</td>
<td>$X_{1,4}$</td>
</tr>
<tr>
<td>$e_2$</td>
<td>$X_{2,1}$</td>
<td>$X_{2,2}$</td>
<td>$X_{2,3}$</td>
<td>$X_{2,4}$</td>
</tr>
<tr>
<td>$e_3$</td>
<td>$X_{3,1}$</td>
<td>$X_{3,2}$</td>
<td>$X_{3,3}$</td>
<td>$X_{3,4}$</td>
</tr>
</tbody>
</table>
Mutation and Crossover

- Mutation of $x_{i,j}$ picks a new dedication uniformly at random from $\{0/k, 1/k, \ldots, k/k\} \setminus x_{i,j}$.

- Crossover: exchange rows

\[
\begin{array}{cccc}
 t_1 & t_2 & t_3 & t_4 \\
 e_1 & X_{1,1} & X_{1,2} & X_{1,3} & X_{1,4} \\
 e_2 & X_{2,1} & X_{2,2} & X_{2,3} & X_{2,4} \\
 e_3 & X_{3,1} & X_{3,2} & X_{3,3} & X_{3,4} \\
\end{array}
\begin{array}{cccc}
 t_1 & t_2 & t_3 & t_4 \\
 e_1 & X_{1,1} & X_{1,2} & X_{1,3} & X_{1,4} \\
 e_2 & X_{2,1} & X_{2,2} & X_{2,3} & X_{2,4} \\
 e_3 & X_{3,1} & X_{3,2} & X_{3,3} & X_{3,4} \\
\end{array}
\]
Mutation and Crossover

- Mutation of $x_{i,j}$ picks a new dedication uniformly at random from $\{0/k, 1/k, \ldots, k/k\} \setminus x_{i,j}$.
- Crossover: exchange rows

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>$X_{1,1}$</td>
<td>$X_{1,2}$</td>
<td>$X_{1,3}$</td>
<td>$X_{1,4}$</td>
</tr>
<tr>
<td>$e_2$</td>
<td>$X_{2,1}$</td>
<td>$X_{2,2}$</td>
<td>$X_{2,3}$</td>
<td>$X_{2,4}$</td>
</tr>
<tr>
<td>$e_3$</td>
<td>$X_{3,1}$</td>
<td>$X_{3,2}$</td>
<td>$X_{3,3}$</td>
<td>$X_{3,4}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>$X_{1,1}$</td>
<td>$X_{1,2}$</td>
<td>$X_{1,3}$</td>
<td>$X_{1,4}$</td>
</tr>
<tr>
<td>$e_2$</td>
<td>$X_{2,1}$</td>
<td>$X_{2,2}$</td>
<td>$X_{2,3}$</td>
<td>$X_{2,4}$</td>
</tr>
<tr>
<td>$e_3$</td>
<td>$X_{3,1}$</td>
<td>$X_{3,2}$</td>
<td>$X_{3,3}$</td>
<td>$X_{3,4}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>$X_{1,1}$</td>
<td>$X_{1,2}$</td>
<td>$X_{1,3}$</td>
<td>$X_{1,4}$</td>
</tr>
<tr>
<td>$e_2$</td>
<td>$X_{2,1}$</td>
<td>$X_{2,2}$</td>
<td>$X_{2,3}$</td>
<td>$X_{2,4}$</td>
</tr>
<tr>
<td>$e_3$</td>
<td>$X_{3,1}$</td>
<td>$X_{3,2}$</td>
<td>$X_{3,3}$</td>
<td>$X_{3,4}$</td>
</tr>
</tbody>
</table>
Mutation and Crossover

- Mutation of $x_{i,j}$ picks a new dedication uniformly at random from \{0/k, 1/k, ..., k/k\} \setminus x_{i,j}.
- Crossover: exchange rows

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>$X_1,1$</td>
<td>$X_1,2$</td>
<td>$X_1,3$</td>
<td>$X_1,4$</td>
</tr>
<tr>
<td>$e_2$</td>
<td>$X_2,1$</td>
<td>$X_2,2$</td>
<td>$X_2,3$</td>
<td>$X_2,4$</td>
</tr>
<tr>
<td>$e_3$</td>
<td>$X_3,1$</td>
<td>$X_3,2$</td>
<td>$X_3,3$</td>
<td>$X_3,4$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>$X_1,1$</td>
<td>$X_1,2$</td>
<td>$X_1,3$</td>
<td>$X_1,4$</td>
</tr>
<tr>
<td>$e_2$</td>
<td>$X_2,1$</td>
<td>$X_2,2$</td>
<td>$X_2,3$</td>
<td>$X_2,4$</td>
</tr>
<tr>
<td>$e_3$</td>
<td>$X_3,1$</td>
<td>$X_3,2$</td>
<td>$X_3,3$</td>
<td>$X_3,4$</td>
</tr>
</tbody>
</table>
Mutation and Crossover

• Mutation of $x_{i,j}$ picks a new dedication uniformly at random from $\{0/k, 1/k, \ldots, k/k\} \setminus x_{i,j}$.

• Crossover: exchange rows or columns.
Designing an Evolutionary Algorithm

- Representation / encoding;
- mutation and crossover;
- fitness / objectives evaluation;
- how to deal with constraints.
Formulating the SPSP

Setting: assume we are given
- \( n \) employees \( e_1, \ldots, e_n \) with salaries and sets of skills;
- \( m \) tasks \( t_1, \ldots, t_m \) with efforts and sets of required skills;
- a task precedence graph (TPG).

Problem: allocate employees to tasks so as to:
- minimise cost (total salaries paid) and
- minimise duration (completion time).

Constraints:
- team must have required skills and
- no overwork.
Evaluating a Solution

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>$X_{1,1}$</td>
<td>$X_{1,2}$</td>
<td>$X_{1,3}$</td>
<td>$X_{1,4}$</td>
</tr>
<tr>
<td>$e_2$</td>
<td>$X_{2,1}$</td>
<td>$X_{2,2}$</td>
<td>$X_{2,3}$</td>
<td>$X_{2,4}$</td>
</tr>
<tr>
<td>$e_3$</td>
<td>$X_{3,1}$</td>
<td>$X_{3,2}$</td>
<td>$X_{3,3}$</td>
<td>$X_{3,4}$</td>
</tr>
</tbody>
</table>

+ TPG, tasks required efforts

+ salaries

Cost and duration
Example for 1 employee, 4 tasks, k = 2

t₁'s required effort and skills: 4 p-month, {sql, java}
t₂'s required effort and skills: 4 p-month, {java}
t₃'s required effort and skills: 8 p-month, {java}
t₄'s required effort and skills: 2 p-month, {java}
e₁'s salary and skills: $1000 per full time month, {sql, java}
Example for 1 employee, 4 tasks, $k = 2$

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

$t_1$'s required effort and skills: 4 p-month, {sql, java}
$t_2$'s required effort and skills: 4 p-month, {java}
$t_3$'s required effort and skills: 8 p-month, {java}
$t_4$'s required effort and skills: 2 p-month, {java}
$e_1$'s salary and skills: $1000 per full time month, {sql, java}

$t_1$: $\frac{4}{0.5} = 8$ months
Example for 1 employee, 4 tasks, $k = 2$

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

$t_1$'s required effort and skills: 4 p-month, \{sql, java\}
$t_2$'s required effort and skills: 4 p-month, \{java\}
$t_3$'s required effort and skills: 8 p-month, \{java\}
$t_4$'s required effort and skills: 2 p-month, \{java\}
$e_1$'s salary and skills: $1000$ per full time month, \{sql, java\}

Gantt Chart

$t_1$: $4 / 0.5 = 8$ months

$t_2$: $4 / 0.5 = 8$ months
Example for 1 employee, 4 tasks, $k = 2$

<table>
<thead>
<tr>
<th>Task</th>
<th>Effort</th>
<th>Skills</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>4</td>
<td>{sql, java}</td>
</tr>
<tr>
<td>$t_2$</td>
<td>4</td>
<td>{java}</td>
</tr>
<tr>
<td>$t_3$</td>
<td>8</td>
<td>{java}</td>
</tr>
<tr>
<td>$t_4$</td>
<td>2</td>
<td>{java}</td>
</tr>
</tbody>
</table>

$e_1$'s salary and skills: $1000 per full time month, {sql, java}
What is the completion time of the project?

And the cost?

t_1's required effort and skills: 4 p-month, {sql, java}
t_2's required effort and skills: 4 p-month, {java}
t_3's required effort and skills: 8 p-month, {java}
t_4's required effort and skills: 2 p-month, {java}
e_1's salary and skills: $1000 per full time month, {sql, java}

gantt Chart

t_1: 4 / 0.5 = 8 months

t_2: 4 / 0.5 = 8 months

t_3: 8 / 0.5 = 16 months

t_4: 2 / 0.5 = 4 months
Designing an Evolutionary Algorithm

- Representation / encoding;
- mutation and crossover;
- fitness / objectives evaluation;
- how to deal with constraints.
Formulating the SPSP

Setting: assume we are given
- \( n \) employees \( e_1, \ldots, e_n \) with salaries and sets of skills;
- \( m \) tasks \( t_1, \ldots, t_m \) with efforts and sets of required skills;
- a task precedence graph (TPG).

Problem: allocate employees to tasks so as to:
- minimise cost (total salaries paid) and
- minimise duration (completion time).

Constraints:
- team must have required skills and
- no overwork.
Infeasible Schedule -- Missing Skills

- **t₁**'s required effort and skills: 4 p-month, **{sql, java}**
- **t₂**'s required effort and skills: 4 p-month, **{java}**
- **t₃**'s required effort and skills: 8 p-month, **{java}**
- **t₄**'s required effort and skills: 2 p-month, **{java}**
- **e₁**'s salary and skills: $1000 per full time month, **{java}**

No. skills missing: 1
Infeasible Schedule -- Missing Skills

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

$t_1$'s required effort and skills: 4 p-month, \{sql, java\}
$t_2$'s required effort and skills: 4 p-month, \{java\}
$t_3$'s required effort and skills: 8 p-month, \{tcp/ip, sql\}
$t_4$'s required effort and skills: 2 p-month, \{java\}
$e_1$'s salary and skills: $1000 per full time month, \{java\}

No. skills missing: 3

Gantt Chart
Overwork

• There is overwork at time $\tau$ if, for a given employee $e_i$, the total dedication of $e_i$ to tasks at time $\tau$ is:

$$\sum_{j \text{ active at } \tau} x_{ij} > 1$$

• Overwork for employee $e_i$ at time $\tau =$

$$\max(0, \sum_{j \text{ active at } \tau} x_{ij} - 1)$$
Infeasible Schedule -- Overwork ($\sum_{j \text{ active at } \tau} X_{ij} > 1$)

t_1: $4 / 0.5 = 8$ months

t_1's required effort and skills: 4 p-month, \{sql, java\}
t_2's required effort and skills: 4 p-month, \{java\}
t_3's required effort and skills: 8 p-month, \{java\}
t_4's required effort and skills: 2 p-month, \{java\}
e_1's salary and skills: $1000$ per full time month, \{sql, java\}

<table>
<thead>
<tr>
<th></th>
<th>t_1</th>
<th>t_2</th>
<th>t_3</th>
<th>t_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_1</td>
<td>0.5</td>
<td>1</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>
Infeasible Schedule -- Overwork ($\sum_{j active at \tau} x_{ij} > 1$)

- $t_1$: 4 / 0.5 = 8 months
- $t_2$: 4 / 1 = 4 months
- $t_3$: 8 / 0.5 = 16 months

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>0.5</td>
<td>1</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

$t_1$'s required effort and skills: 4 p-month, \{sql, java\}
$t_2$'s required effort and skills: 4 p-month, \{java\}
$t_3$'s required effort and skills: 8 p-month, \{java\}
$t_4$'s required effort and skills: 2 p-month, \{java\}
$e_1$'s salary and skills: $1000 per full time month, \{sql,java\}
Infeasible Schedule --

Overwork ($\sum_j a_{\text{active at } \tau} x_{ij} > 1$)

t₁'s required effort and skills: 4 p-month, {sql, java}
t₂'s required effort and skills: 4 p-month, {java}
t₃'s required effort and skills: 8 p-month, {java}
t₄'s required effort and skills: 2 p-month, {java}
e₁'s salary and skills: $1000 per full time month, {sql, java}

<table>
<thead>
<tr>
<th></th>
<th>t₁</th>
<th>t₂</th>
<th>t₃</th>
<th>t₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>e₁</td>
<td>0.5</td>
<td>1</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Gantt Chart

- t₁: $4 / 0.5 = 8$ months
- t₂: $4 / 1 = 4$ months
- t₃: $8 / 0.5 = 16$ months
- t₄: $2 / 0.5 = 4$ months

$\sum_j a_{\text{active at } \tau} x_{1j} = 1.5$
How to Deal with the Constraints?

• Option 1: assign very high cost and duration if constraints are violated.

Cost = $n_{cost\_penal} + n_{over} \times overwork + n_{skills} \times skills\_missing$

Duration = $n_{dur\_penal} + n_{over} \times overwork + n_{skills} \times skills\_missing$

- Value higher than any feasible cost and duration.
- Total amount of overwork time.
- Total number of skills missing in the teams.
How to Deal with the Constraints?

• Option 1: assign very high cost and duration if constraints are violated.

Cost = $n_{cost\_penal} + n_{over} \times overwork + n_{skills} \times skills\_missing$

Duration = $n_{dur\_penal} + n_{over} \times overwork + n_{skills} \times skills\_missing$

What are the problems of this solution?
How to Deal with the Constraints?

• Option 2: normalise dedications to deal with overwork so that total dedication is at most 1.

If employee $i$ has overwork at any moment $\tau$

$$d_{ij}(\tau) = \frac{x_{ij}}{\sum_{j \text{ active at } \tau} x_{ij}}$$

else

$$d_{ij}(\tau) = x_{ij}$$

Cost = $n_{\text{cost_penal}} \times 2 \times \text{skills_missing}$

Duration = $n_{\text{dur_penal}} \times 2 \times \text{skills_missing}$
Infeasible Schedule -- Overwork ($\sum_{j \text{ active at } \tau} x_{ij} > 1$)

$t_1$: $4 / 0.5 = 8$ months
$t_2$: $4 / 1 = 4$ months
$t_3$: $8 / 0.5 = 16$ months

$t_1$'s required effort and skills: 4 p-month, \{sql, java\}
$t_2$'s required effort and skills: 4 p-month, \{java\}
$t_3$'s required effort and skills: 8 p-month, \{java\}
$t_4$'s required effort and skills: 2 p-month, \{java\}
$e_1$'s salary and skills: $1000$ per full time month, \{sql, java\}
Infeasible Schedule -- Overwork ($\sum_{j \text{ active at } \tau} x_{ij} > 1$)

$t_1$: 4 / 0.5 = 8 months

$t_2$: 4 / (1/1.5) = 6 months

$t_3 = 8 / (0.5/1.5) = 24$ months

$t_4$'s required effort and skills: 2 p-month, {java}

e_1's salary and skills: $1000 per full time month, {sql, java}

t_1's required effort and skills: 4 p-month, {sql, java}
t_2's required effort and skills: 4 p-month, {java}
t_3's required effort and skills: 8 p-month, {java}

\[ d_{1,2} = 1/1.5 \]
\[ d_{1,3} = 0.5/1.5 \]
Infeasible Schedule --
Overwork ($\sum_{j \text{ active at } \tau} x_{ij} > 1$)

$t_1$: $4 / 0.5 = 8$ months

$t_2$: $4 / (1/1.5) = 6$ months

$t_3$: 6 months with $d_{13} = 0.5/1.5$

$t_3$: $(8 - 2) / 0.5 = 12$ months

$t_4$: (Graph or table)

$e_1$: $0.5 \quad 1 \quad 0.5 \quad 0.5$

$t_1$'s required effort and skills: 4 p-month, \{sql, java\}
$t_2$'s required effort and skills: 4 p-month, \{java\}
$t_3$'s required effort and skills: 8 p-month, \{java\}
$t_4$'s required effort and skills: 2 p-month, \{java\}
$e_1$'s salary and skills: $1000$ per full time month, \{sql,java\}

$= 2$ person-month
Infeasible Schedule -- Overwork ($\sum_{j \text{ active at } \tau} x_{ij} > 1$)

t_1: 4 / 0.5 = 8 months

t_2: 4 / (1/1.5) = 6 months

t_3: 6 months with $d_{13} = 0.5/1.5$

t_4: 2 / 0.5 = 4 months

t_1's required effort and skills: 4 p-month, {sql, java}
t_2's required effort and skills: 4 p-month, {java}
t_3's required effort and skills: 8 p-month, {java}
t_4's required effort and skills: 2 p-month, {java}
e_1's salary and skills: $1000 per full time month, {sql, java}
[Demo -- Ian Watson's final year project (2013)]
Summary

• What Software Project Scheduling Problem (SPSP) is.
• Why are automated optimisation methods are important for the SPSP.
• How to solve the SPSP:
  • suitable EA;
  • representation;
  • mutation and crossover;
  • objectives;
  • constraints.
Further Reading

