Software Project Scheduling Problem

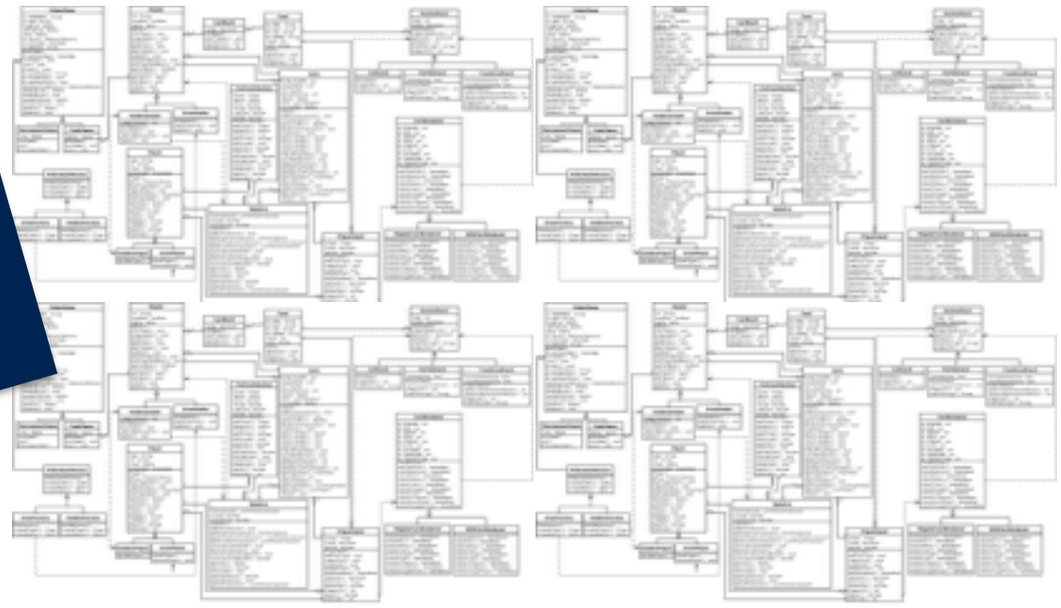
Leandro Minku
www.cs.bham.ac.uk/~minkull

Nature-inspired Optimisation Lecture

Outline

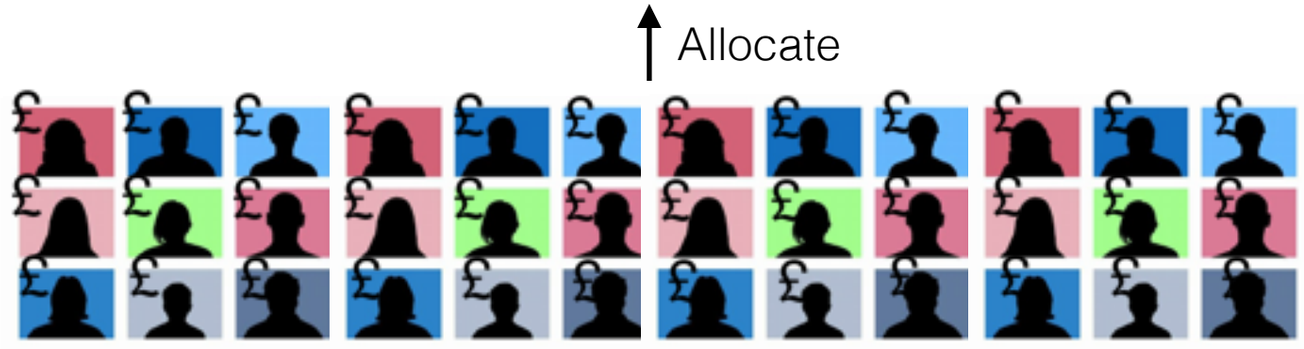
- What is the Software Project Scheduling Problem (SPSP)?
- Why are automated optimisation methods important for the SPSP?
- How to solve the SPSP?

Software Project Scheduling Problem (SPSP)



Each task requires a set of skills and effort.

Tasks also have a precedence relation.



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].

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).

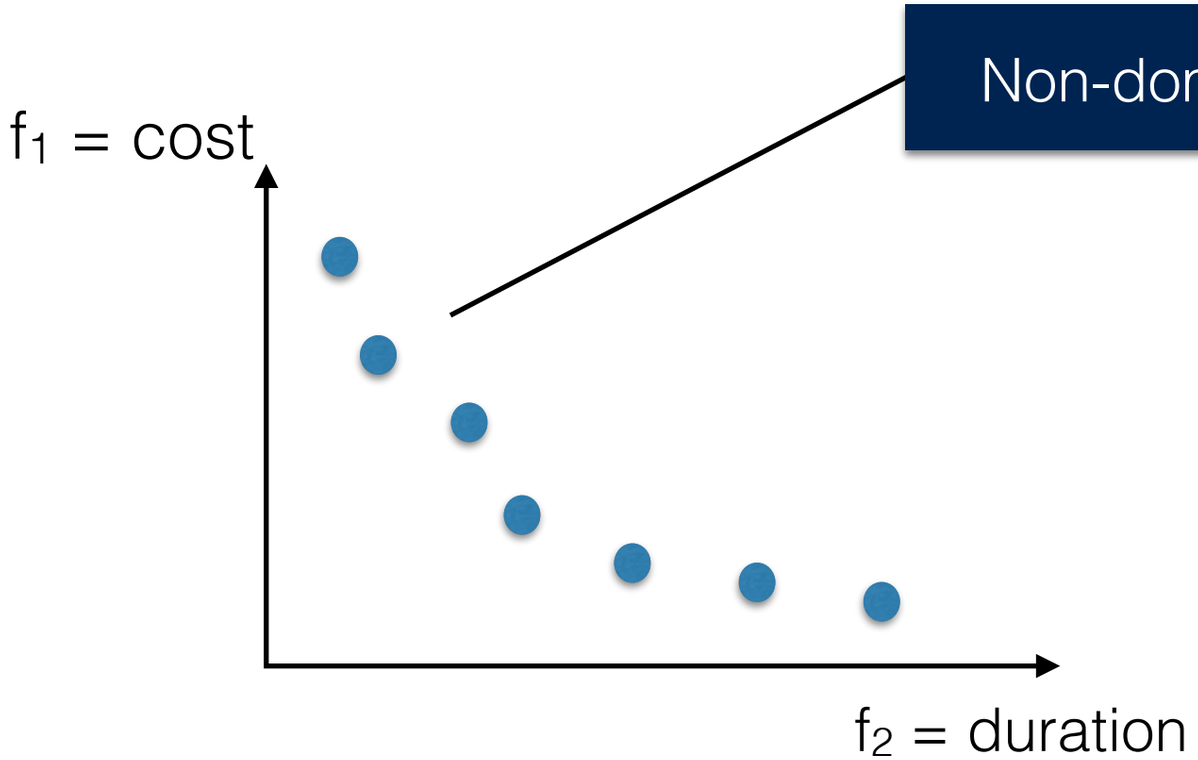
Problem: allocate employees to tasks so as to:
- minimise cost (total salaries paid) and
- minimise duration (completion time).

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

Solving the SPSP

What type of evolutionary algorithm would be adequate?

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



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*.

	t_1	t_2	...	t_m
e_1	$x_{1,1}$	$x_{1,2}$	...	$x_{1,m}$
e_2	$x_{2,1}$	$x_{2,2}$	...	$x_{2,m}$
...	...	...	...	...
e_n	$x_{n,1}$	$x_{n,2}$	...	$x_{n,m}$

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

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, …, k/k} \ $x_{i,j}$.
- Crossover: exchange rows

	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	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$

→

	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	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$

	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	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$

→

	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	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$

Mutation and Crossover

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

	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	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$

→

	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	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$

	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	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$

→

	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	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$

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

	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	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$

	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	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$

	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	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$

	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	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$

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

	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	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$

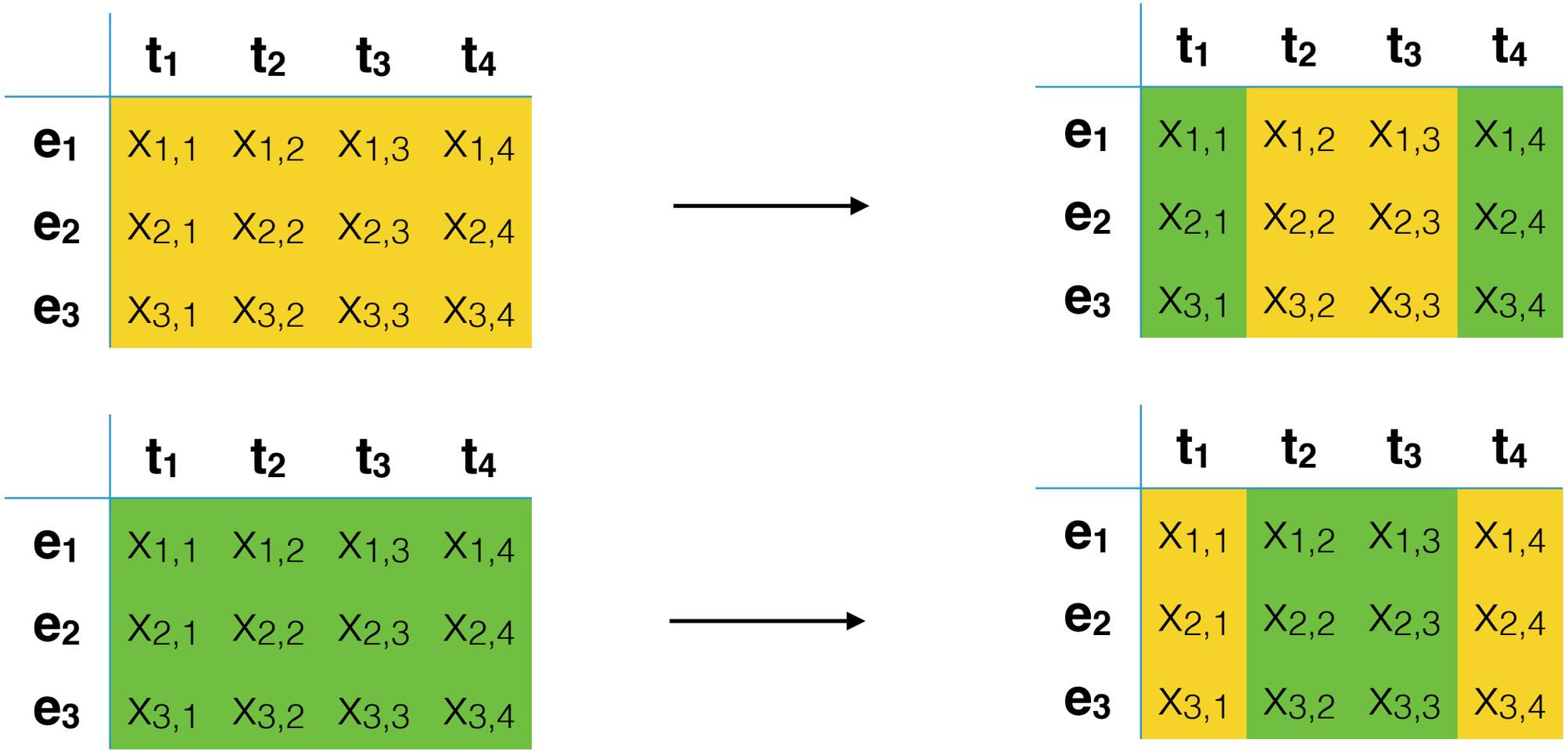
	t_1	t_2	t_3	t_4
e_1	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$
e_2	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{2,4}$
e_3	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$

	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	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$

	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	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$

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).

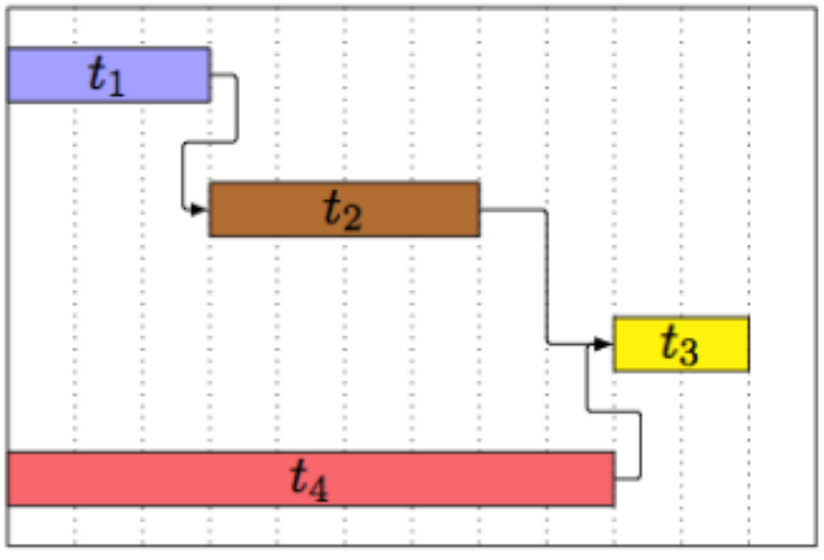
Problem: allocate employees to tasks so as to:
- minimise cost (total salaries paid) and
- minimise duration (completion time).

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

Evaluating a Solution

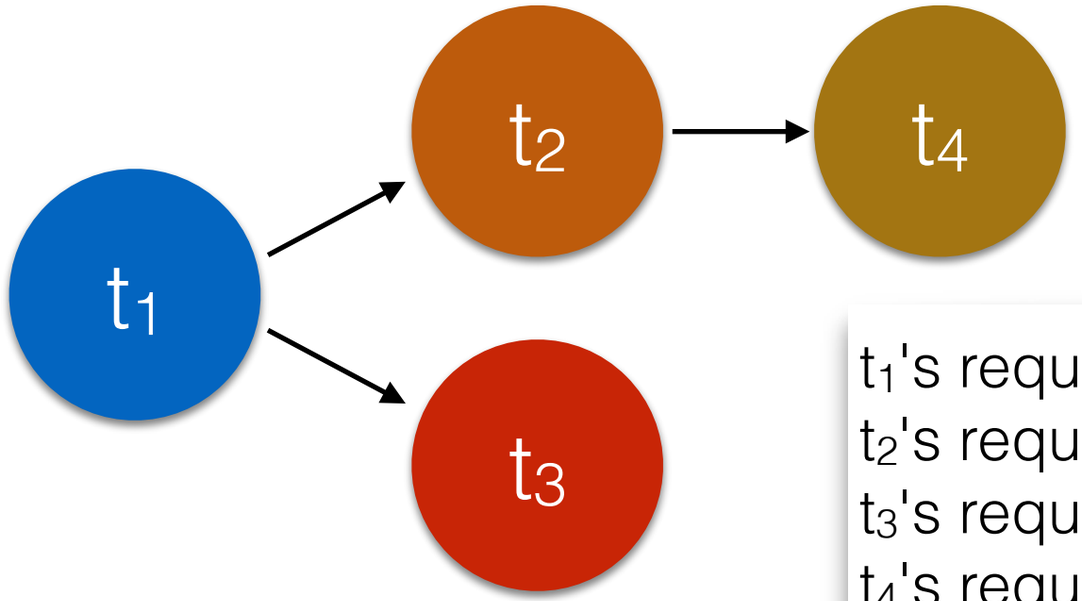
	t_1	t_2	t_3	t_4
e_1	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$
e_2	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{2,4}$
e_3	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$

+ TPG, tasks required efforts



+ salaries → Cost and duration

Example for 1 employee, 4 tasks, k = 2

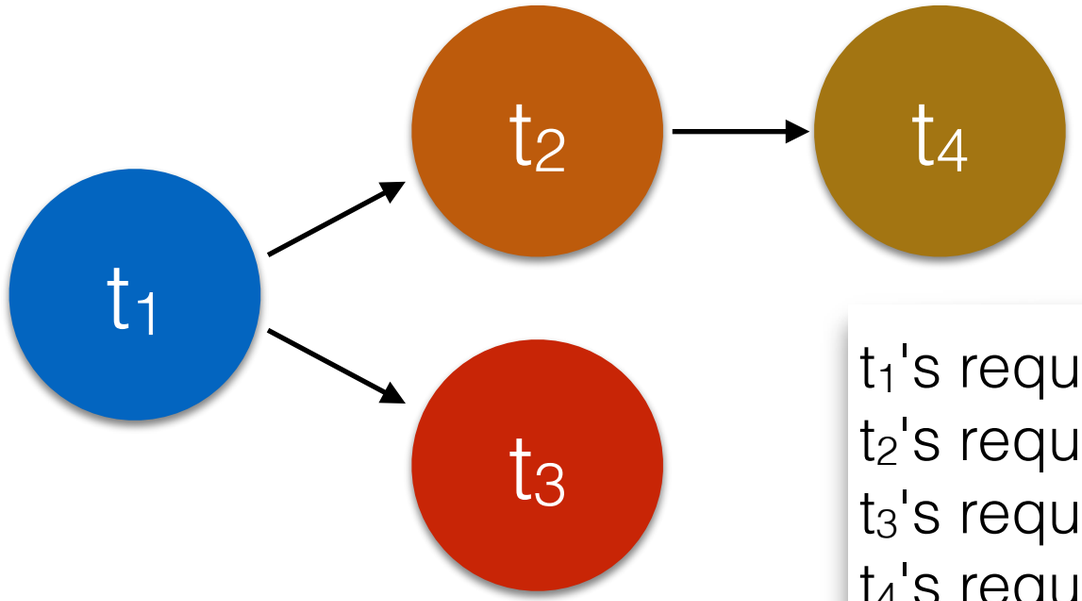


	t1	**t2**	**t3**	**t4**
e1	0.5	0.5	0.5	0.5

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}

Gantt Chart

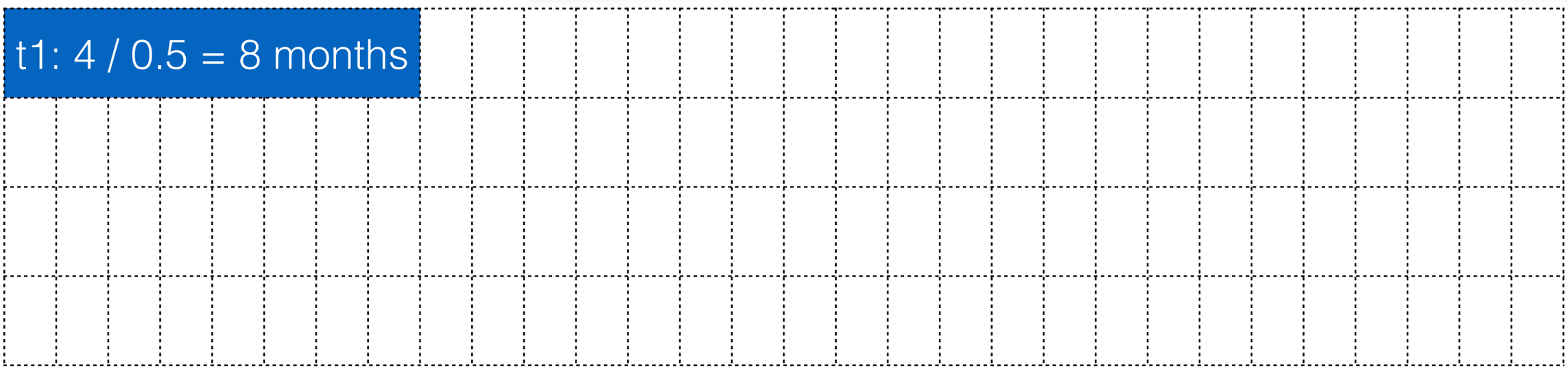
Example for 1 employee, 4 tasks, k = 2



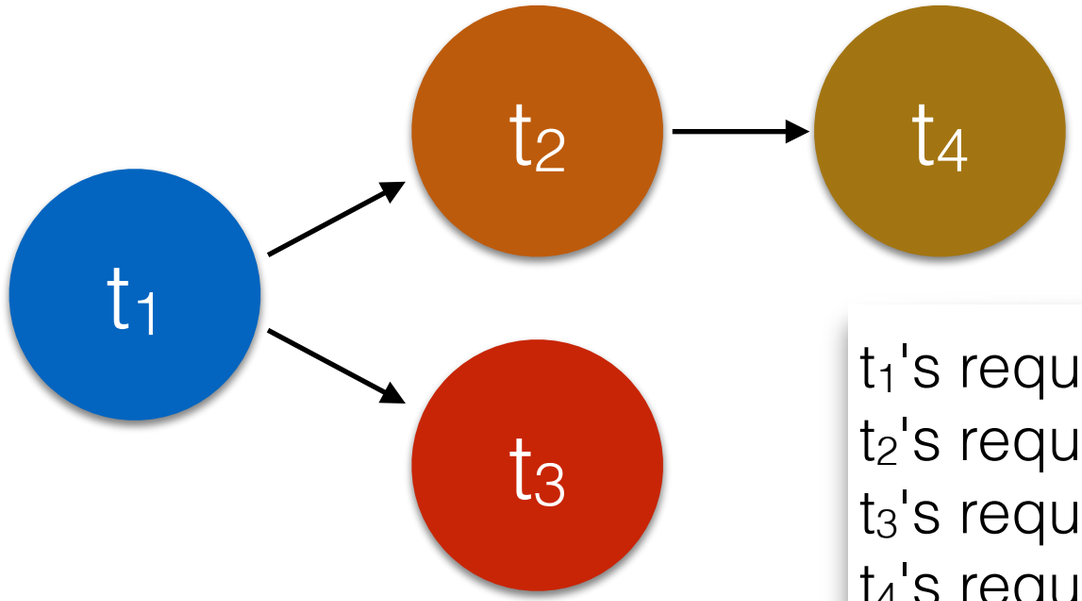
	t1	**t2**	**t3**	**t4**
e1	0.5	0.5	0.5	0.5

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}

Gantt Chart



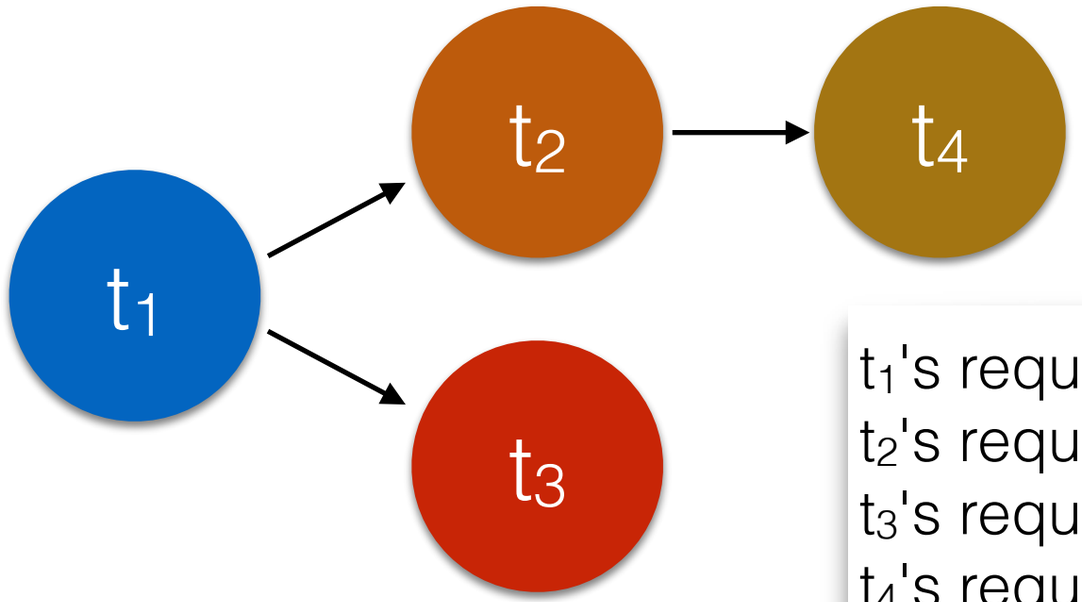
Example for 1 employee, 4 tasks, k = 2



	t1	**t2**	**t3**	**t4**
e1	0.5	0.5	0.5	0.5

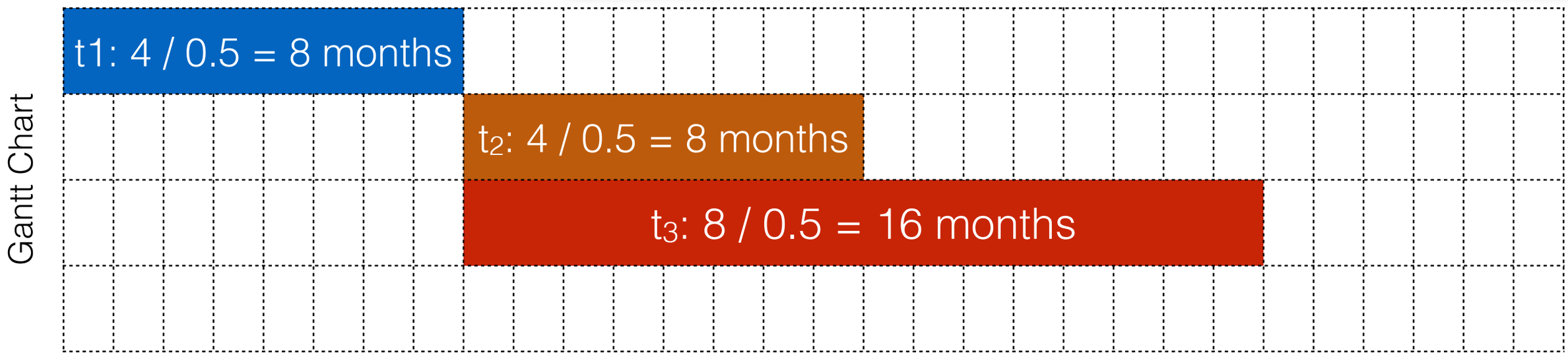
t1's required effort and skills: 4 p-month, {sql, java}
t2's required effort and skills: 4 p-month, {java}
t3's required effort and skills: 8 p-month, {java}
t4's required effort and skills: 2 p-month, {java}
e1's salary and skills: $1000 per full time month, {sql, java}

Example for 1 employee, 4 tasks, k = 2



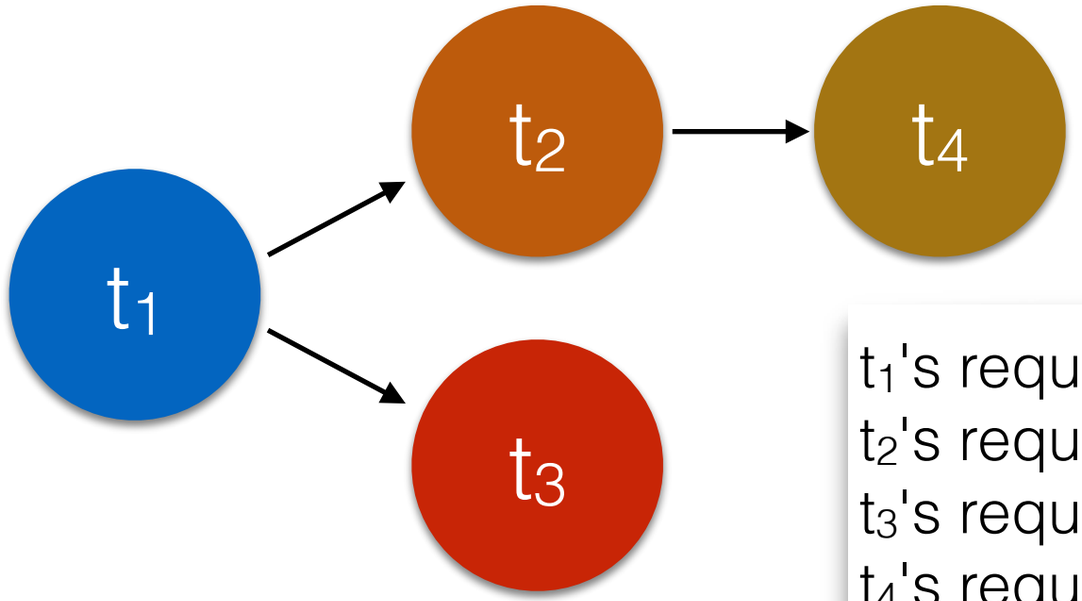
	t_1	t_2	t_3	t_4
e_1	0.5	0.5	0.5	0.5

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}



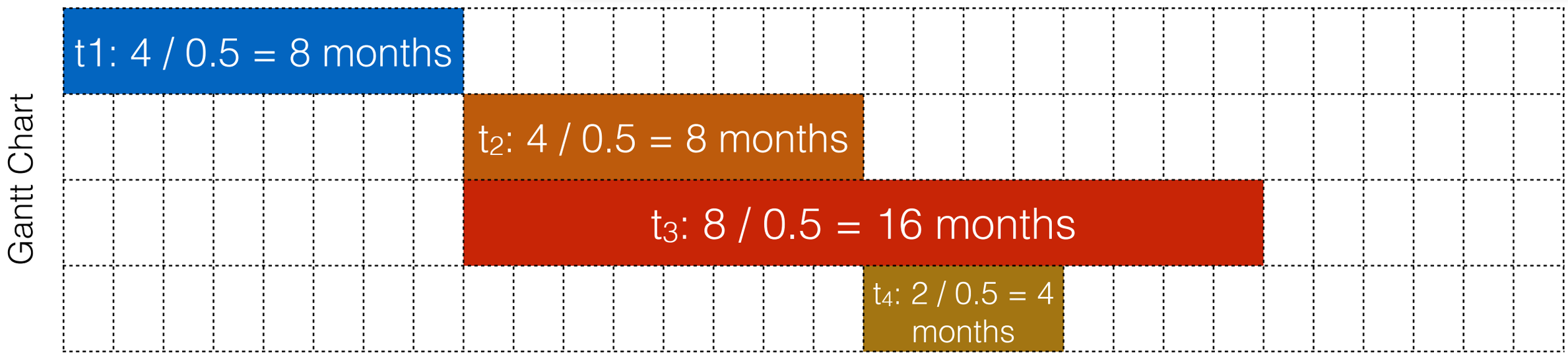
What is the completion time of the project?

And the cost?



	t_1	t_2	t_3	t_4
e_1	0.5	0.5	0.5	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}



Designing an Evolutionary Algorithm

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

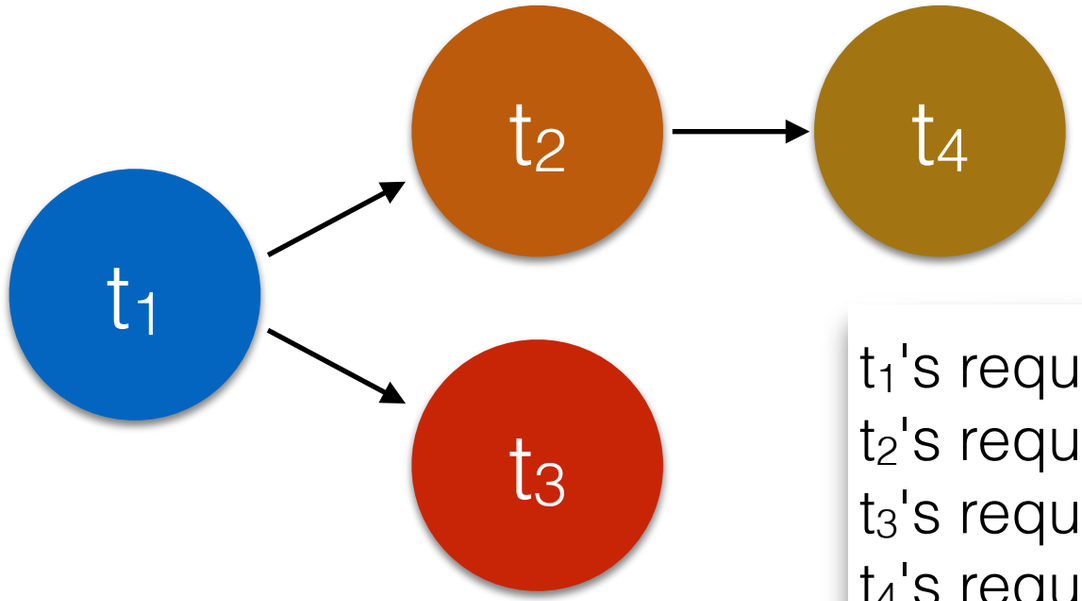
Formulating the SPSP

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).

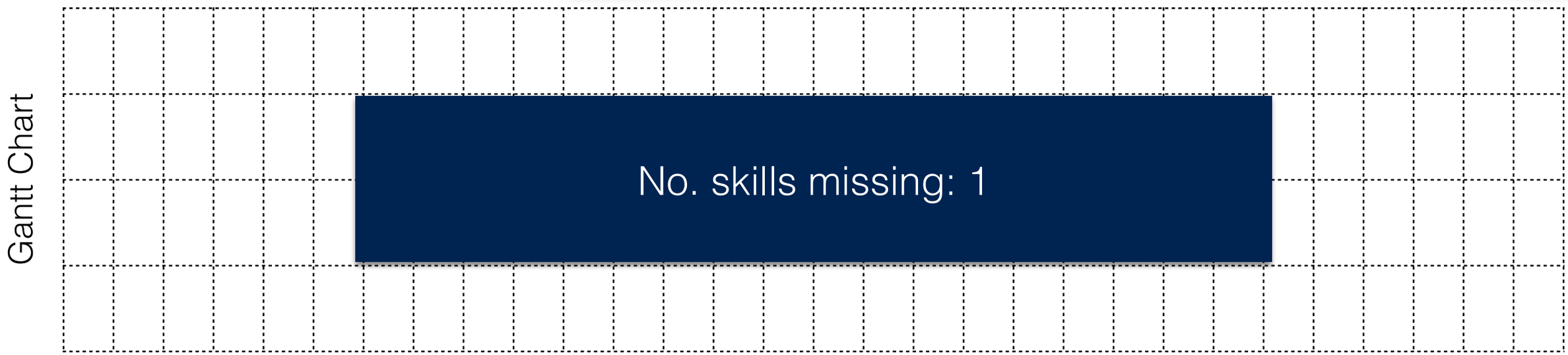
Constraints:
- team must have required skills and
- no overwork.

Infeasible Schedule -- Missing Skills

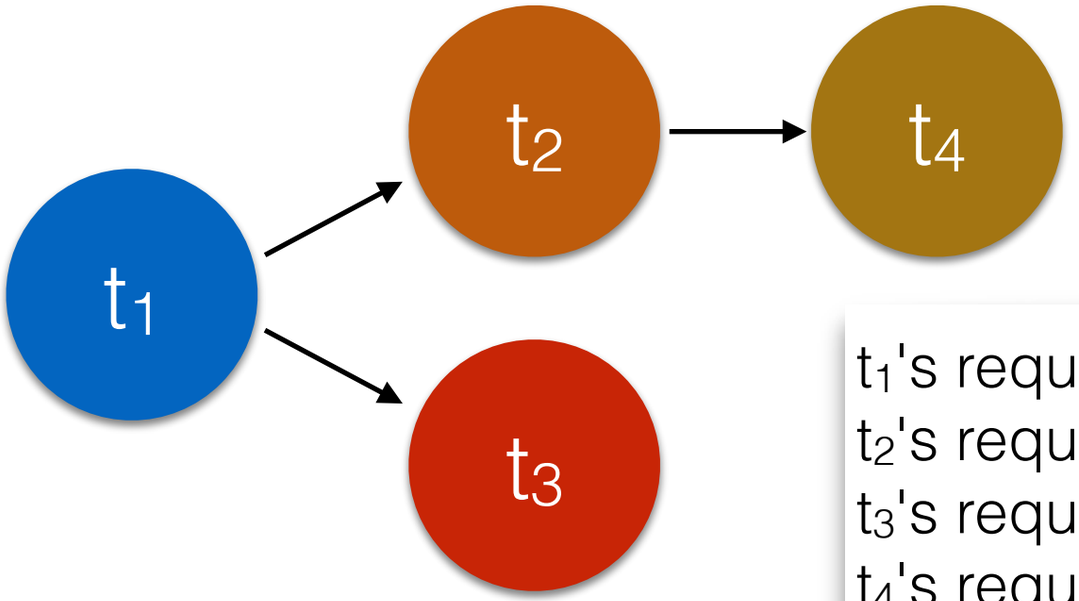


	t_1	t_2	t_3	t_4
e_1	0.5	0.5	0.5	0.5

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, {java}

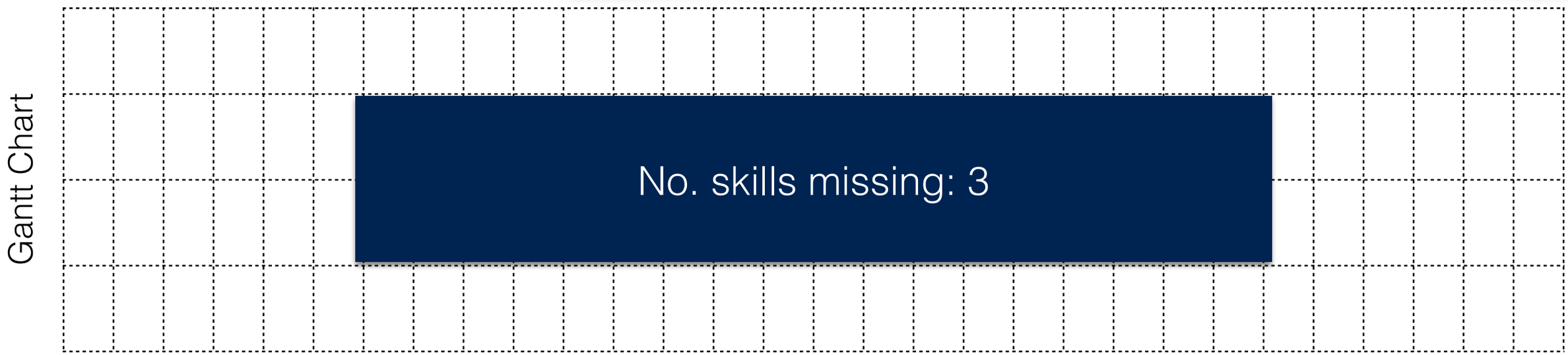


Infeasible Schedule -- Missing Skills



	t_1	t_2	t_3	t_4
e_1	0.5	0.5	0.5	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, {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}



Overwork

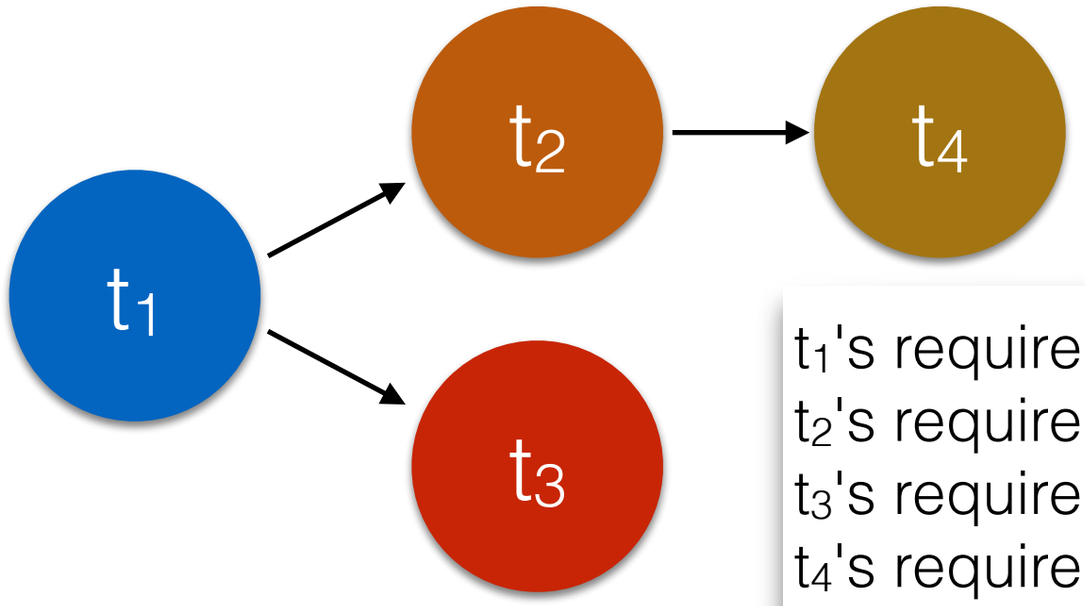
- There is overwork at time τ if, for a given employee e_i, the total dedication of e_i to tasks at time τ is:

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

- Overwork for employee e_i at time τ =

  $\max(0, \sum_{j \text{ active at } \tau} x_{ij} - 1)$

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

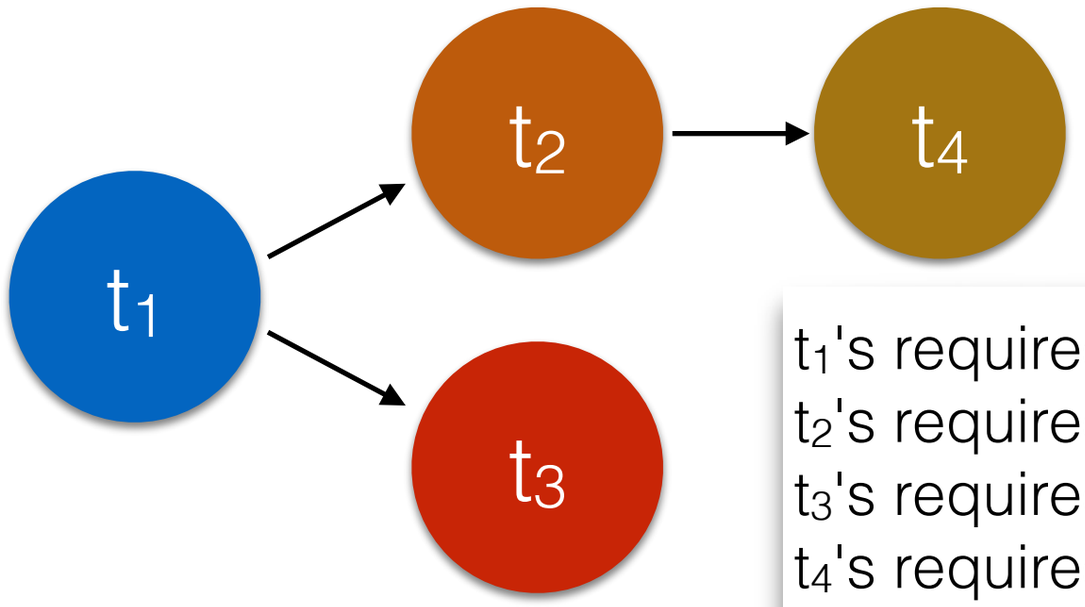


	t_1	t_2	t_3	t_4
e_1	0.5	1	0.5	0.5

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}

Gantt Chart

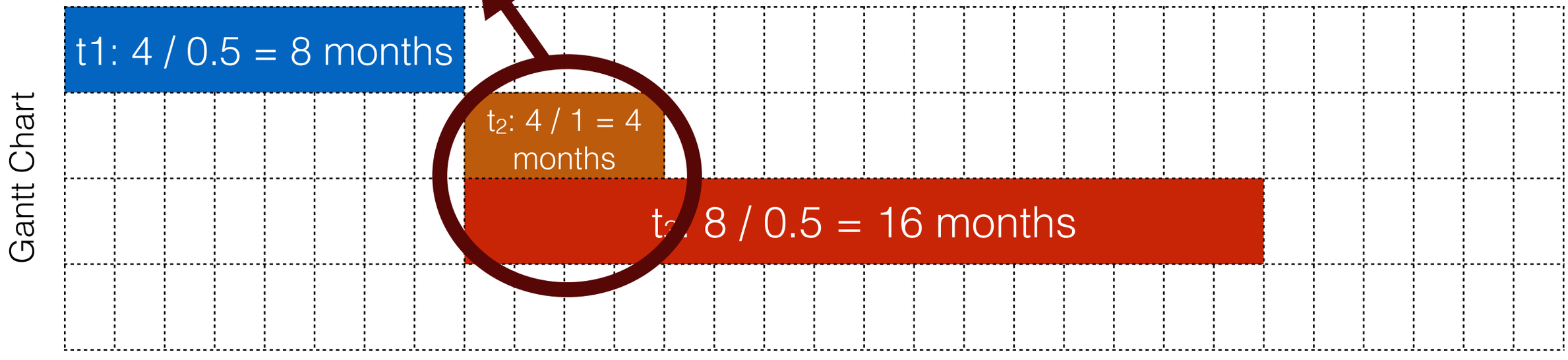
Infeasible Schedule -- Overwork ($\sum_{j \text{ active at } \tau} x_{ij} > 1$)



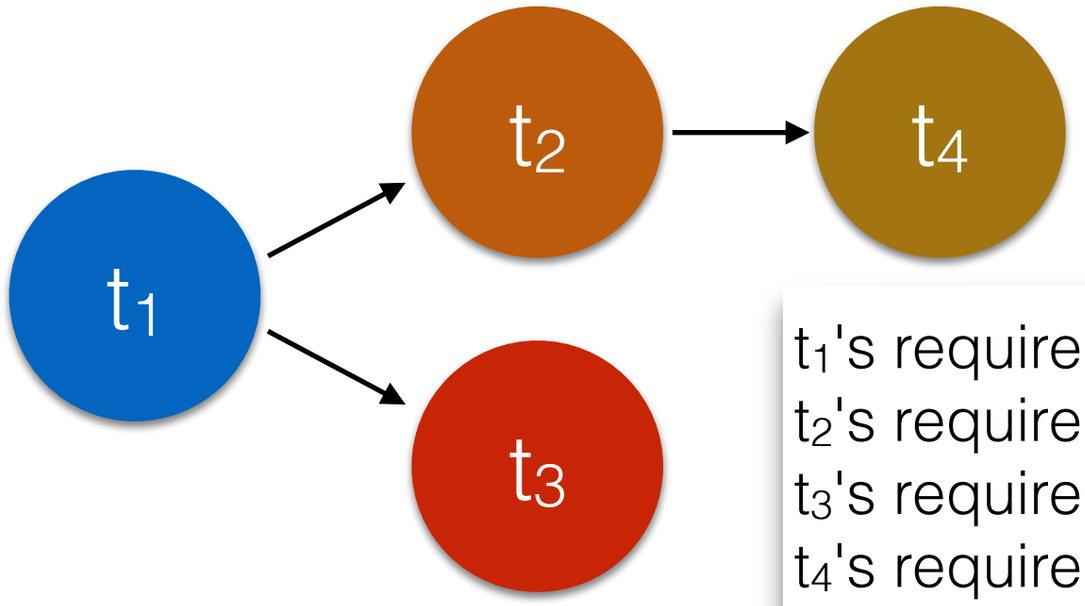
	t_1	t_2	t_3	t_4
e_1	0.5	1	0.5	0.5

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}

$\sum_{j \text{ active at } \tau} x_{1j} = 1.5$



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



	t_1	t_2	t_3	t_4
e_1	0.5	1	0.5	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}

$\sum_{j \text{ active at } \tau} x_{1j} = 1.5$

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

How to Deal with the Constraints?

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

Cost = n_{cost_penal} + n_{over} * overwork + n_{skills} * skills_missing

Duration = n_{dur_penal} + n_{over} * overwork + n_{skills} * 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} * overwork + n_{skills} * skills_missing

  Duration = n_{dur_penal} + n_{over} * overwork + n_{skills} * 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 τ

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

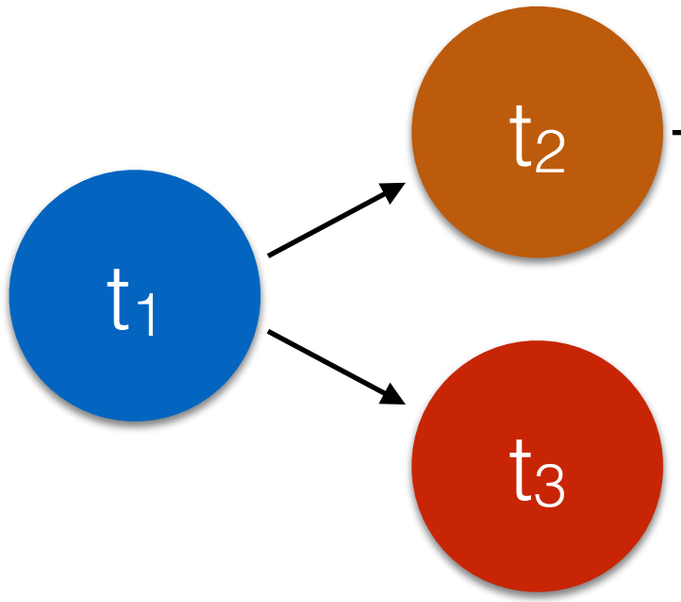
else

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

Cost = n_{cost_penal} * 2 * skills_missing

Duration = n_{dur_penal}* 2 * skills_missing

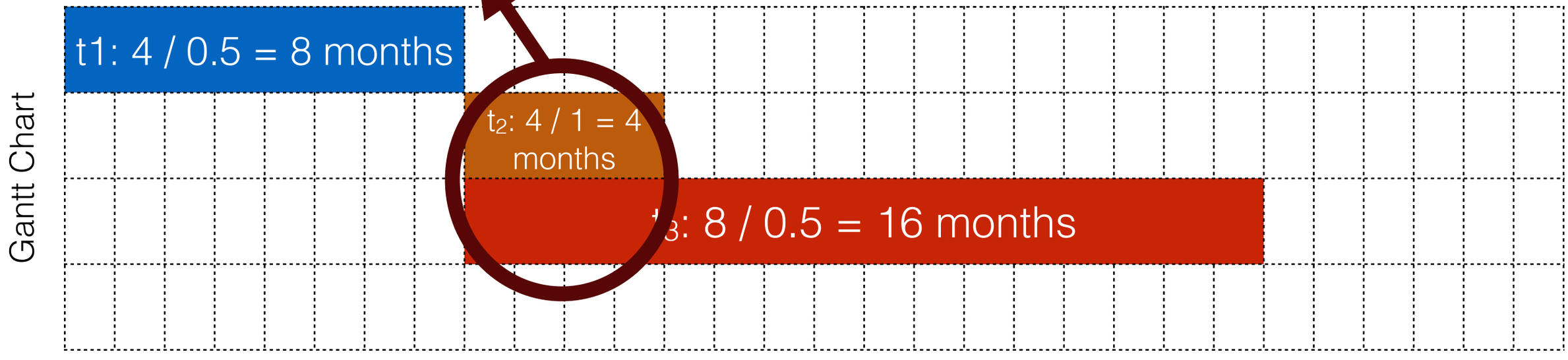
Infeasible Schedule -- Overwork ($\sum_{j \text{ active at } \tau} x_{ij} > 1$)



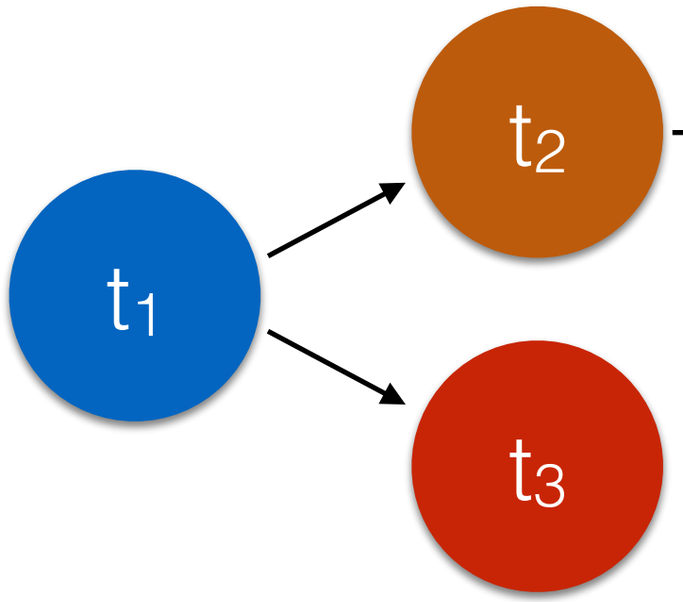
	t_1	t_2	t_3	t_4
e_1	0.5	1	0.5	0.5

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}

$\sum_{j \text{ active at } \tau} x_{1j} = 1.5$



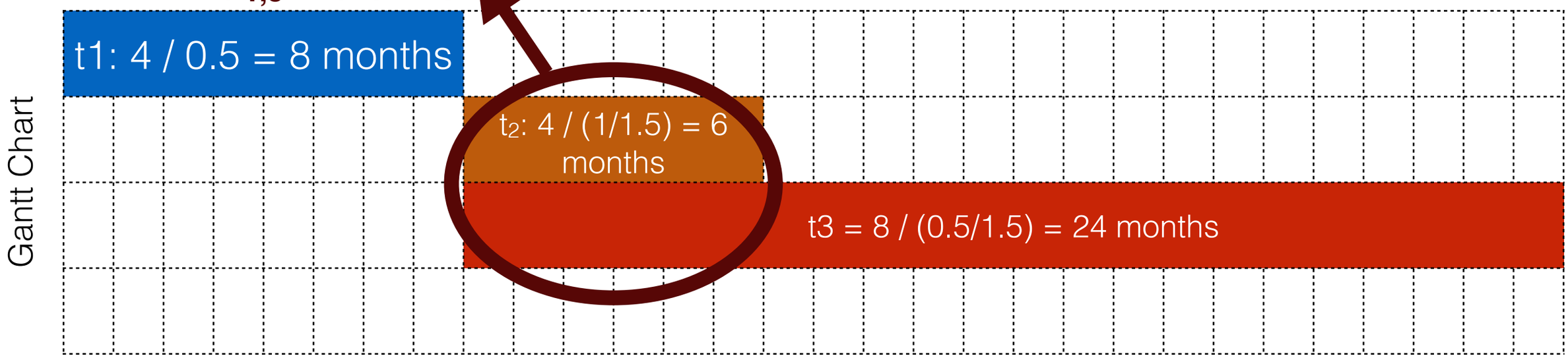
Infeasible Schedule -- Overwork ($\sum_{j \text{ active at } \tau} x_{ij} > 1$)



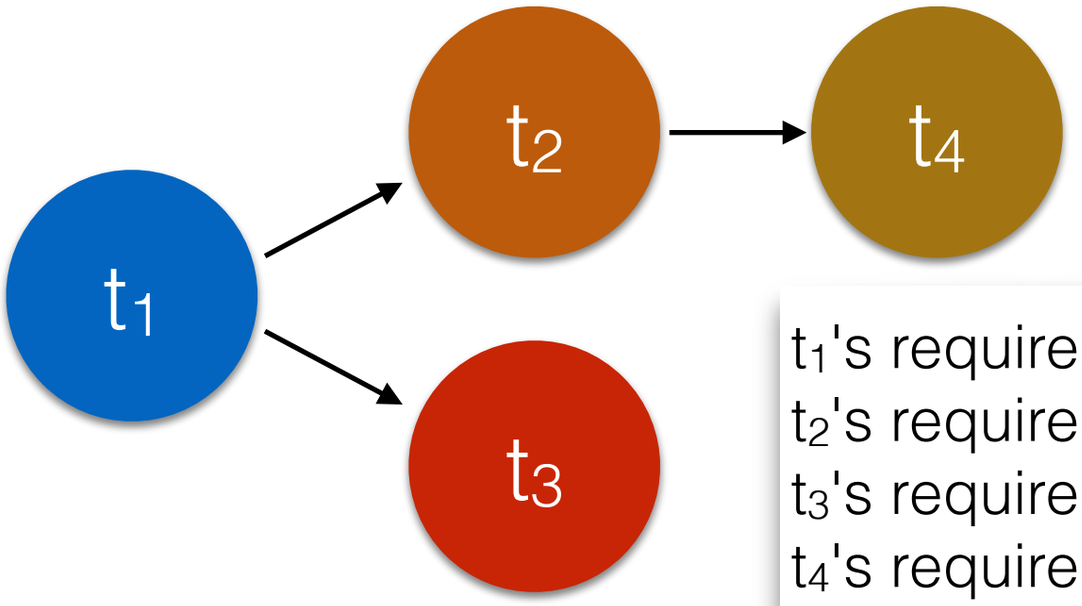
	t_1	t_2	t_3	t_4
e_1	0.5	1	0.5	0.5

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}

$d_{1,2} = 1/1.5$
$d_{1,3} = 0.5/1.5$



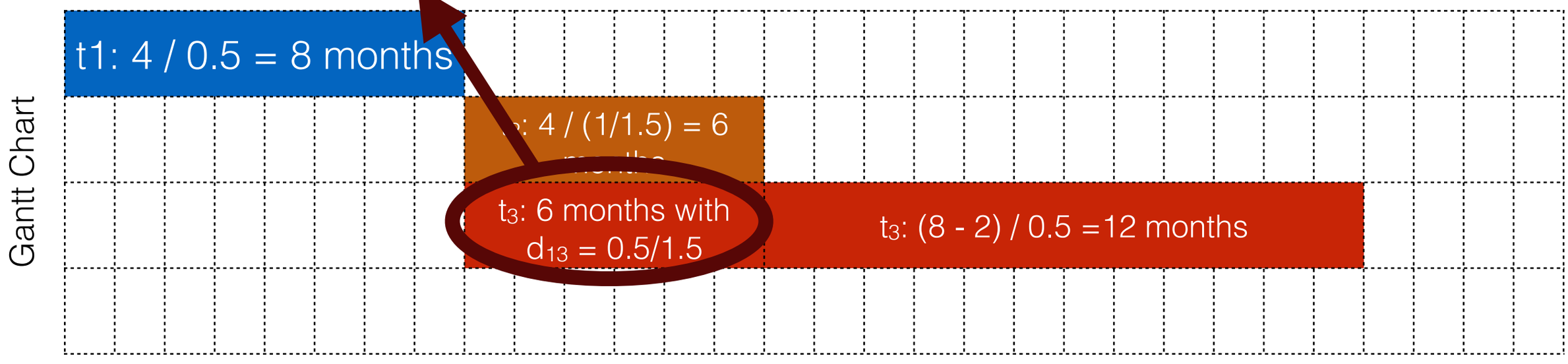
Infeasible Schedule -- Overwork ($\sum_{j \text{ active at } \tau} x_{ij} > 1$)



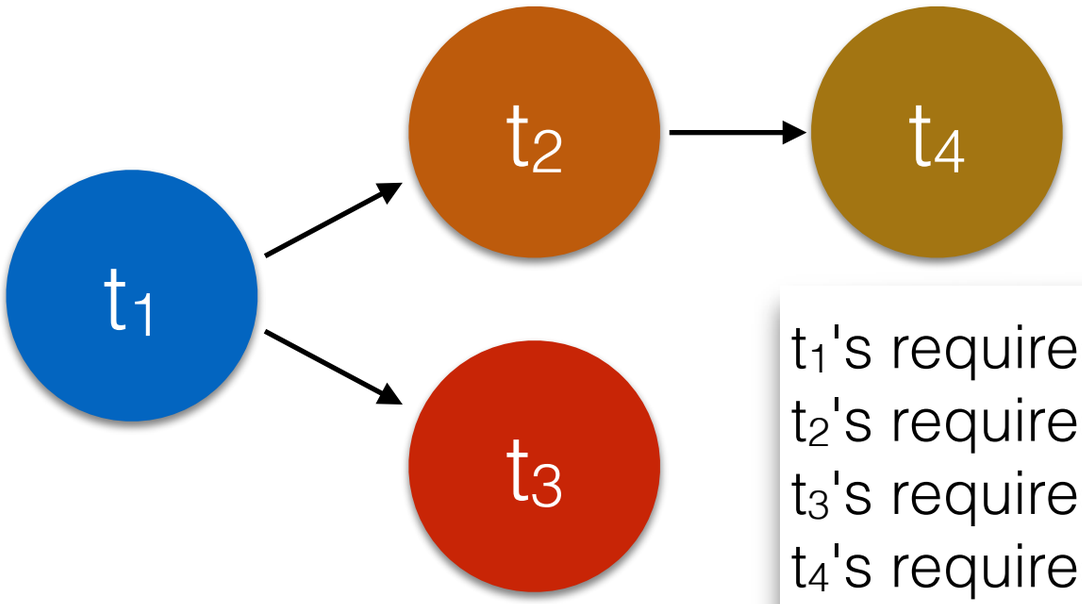
	t_1	t_2	t_3	t_4
e_1	0.5	1	0.5	0.5

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

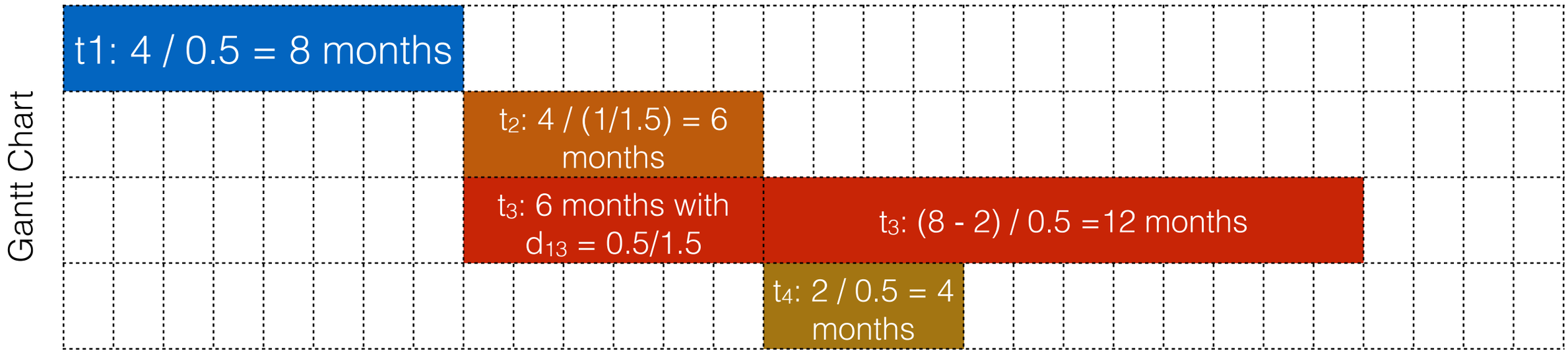


Infeasible Schedule -- Overwork ($\sum_{j \text{ active at } \tau} x_{ij} > 1$)



	t_1	t_2	t_3	t_4
e_1	0.5	1	0.5	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}



[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

L. Minku, D. Sudholt and X. Yao. "Improved Evolutionary Algorithm Design for the Project Scheduling Problem Based on Runtime Analysis", IEEE Transactions on Software Engineering vol. 40, n. 1, p. 83-102, 2014.

F. Luna, D. Gonzalez-Alvarez, F. Chicano, and M. Vega- Rodrıguez, "On the Scalability of Multi-Objective Metaheuristics for the Software Scheduling Problem", Proceedings of the 11th International Conference on Intelligent System Design and Applications, p. 1110-1115, 2011.

E. Alba and F. Chicano, "Software Project Management with GAs", Information Sciences, vol. 177, p. 2380-2401, 2007.