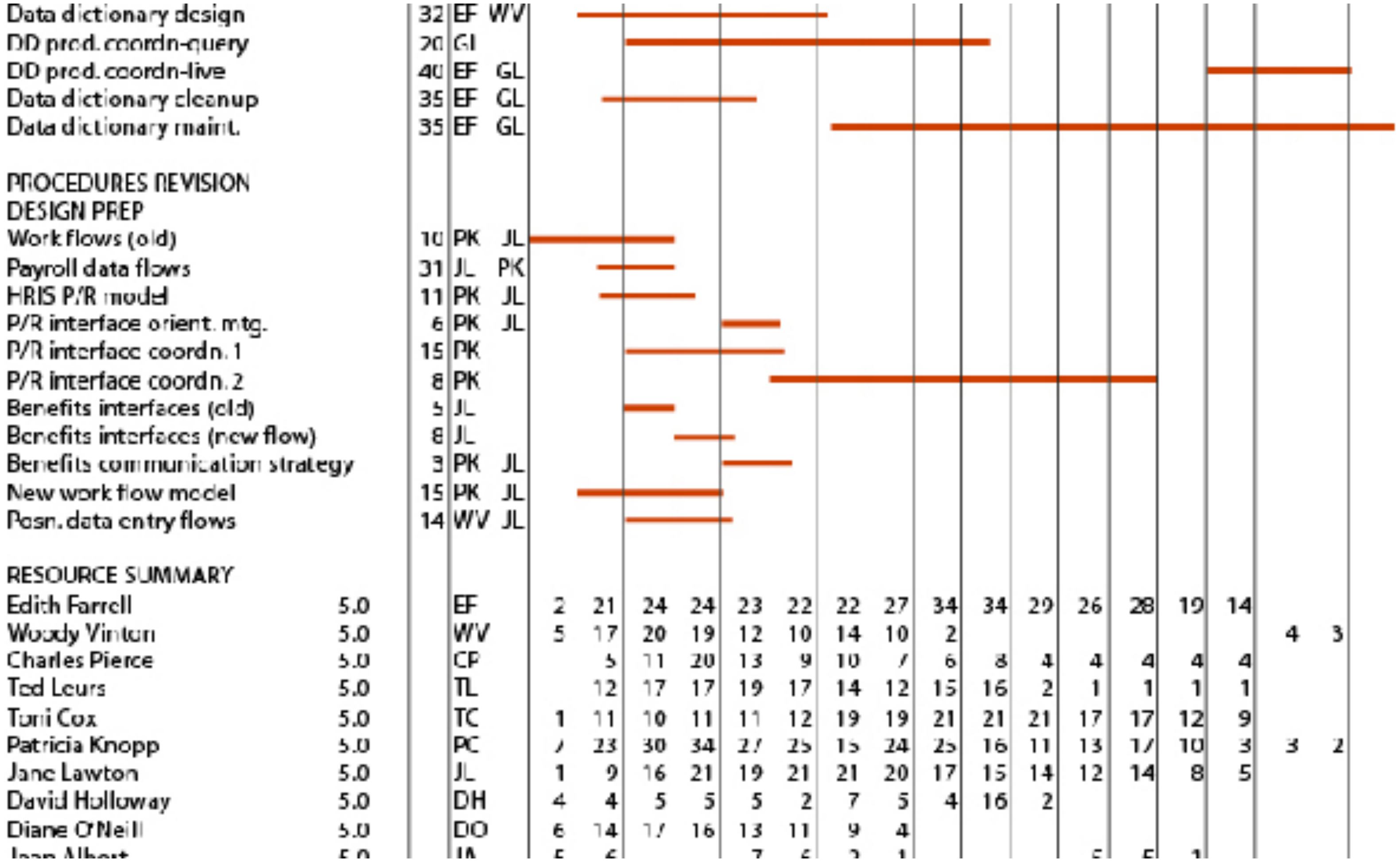
CO3091 - Computational Intelligence and Software Engineering

Lecture 11



Software Project Scheduling — Part II

Leandro L. Minku

Overview

- Part I:
  - What is the Software Project Scheduling Problem (SPSP)?
  - Why are automated methods important for the SPSP?
  - How to formulate the SPSP as an optimisation problem?
  - How to solve the SPSP using optimisation algorithms?
  - [Except constraints]
- Part II:
  - How to formulate the constraints of the SPSP
  - How to deal with the constraints of the SPSP

Summary of Problem Formulation

- Design variable:
  - Matrix of dedications of employees to tasks.
  - Dedications have a granularity *k*.

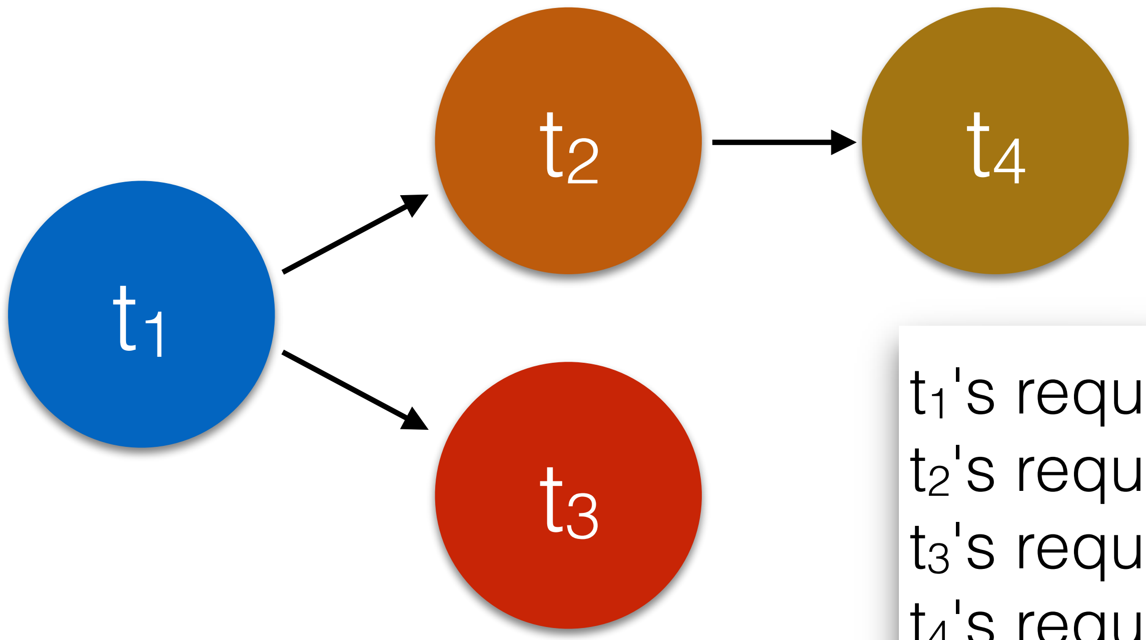
- Objectives:
  - Cost (to be minimised).
  - Duration (to be minimised).

- **Constraints:**
  - **Team of employees must have required skills to perform a given task.**
  - **No overwork.**

Skills Constraint

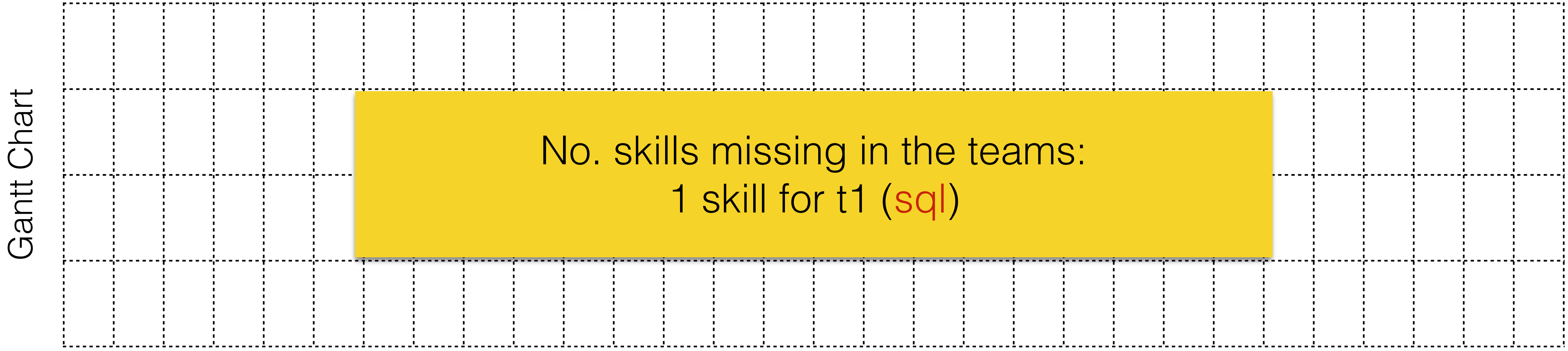
Teams assigned to tasks must have all required skills to perform those tasks.

Example of Infeasible Schedule — Missing Skills



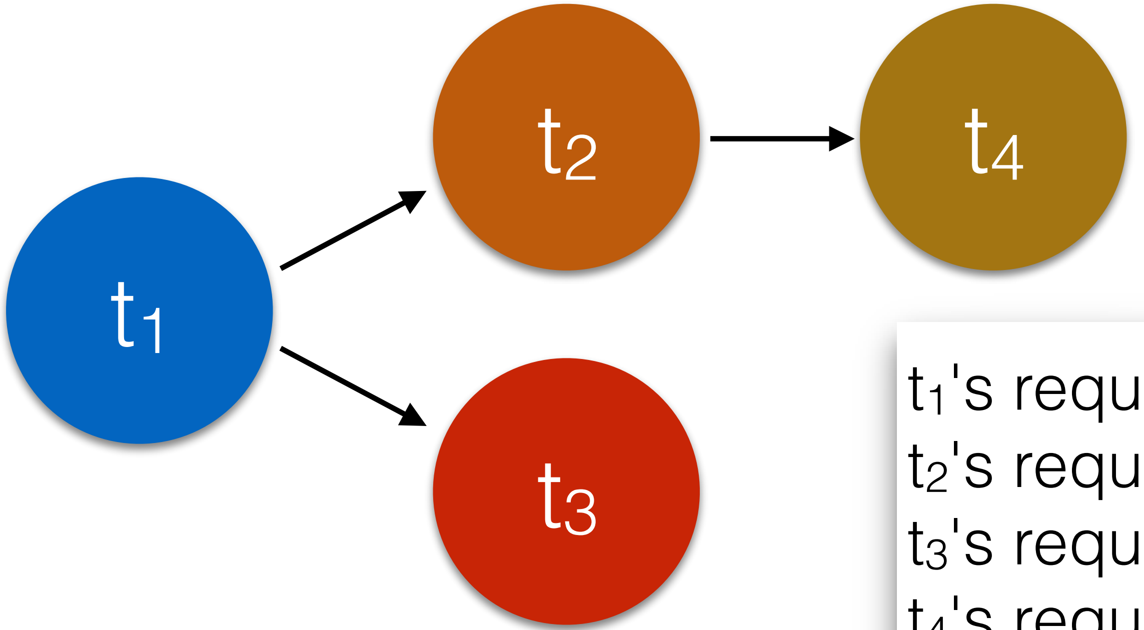
x’	t_1	t_2	t_3	t_4
e_1	0.5	0.5	0.5	0.5

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}



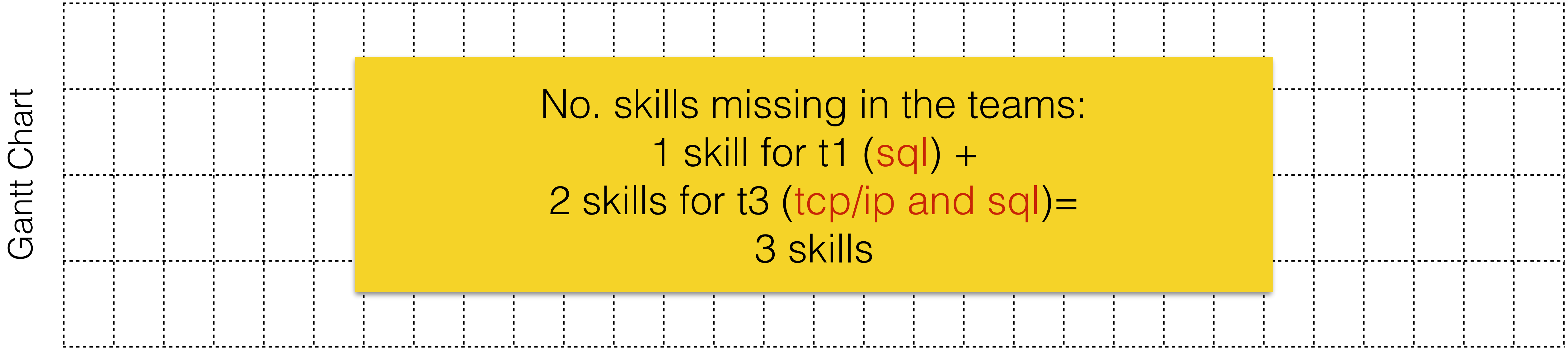
No. skills missing in the teams:
1 skill for t1 (sql)

Example of Infeasible Schedule — Missing Skills



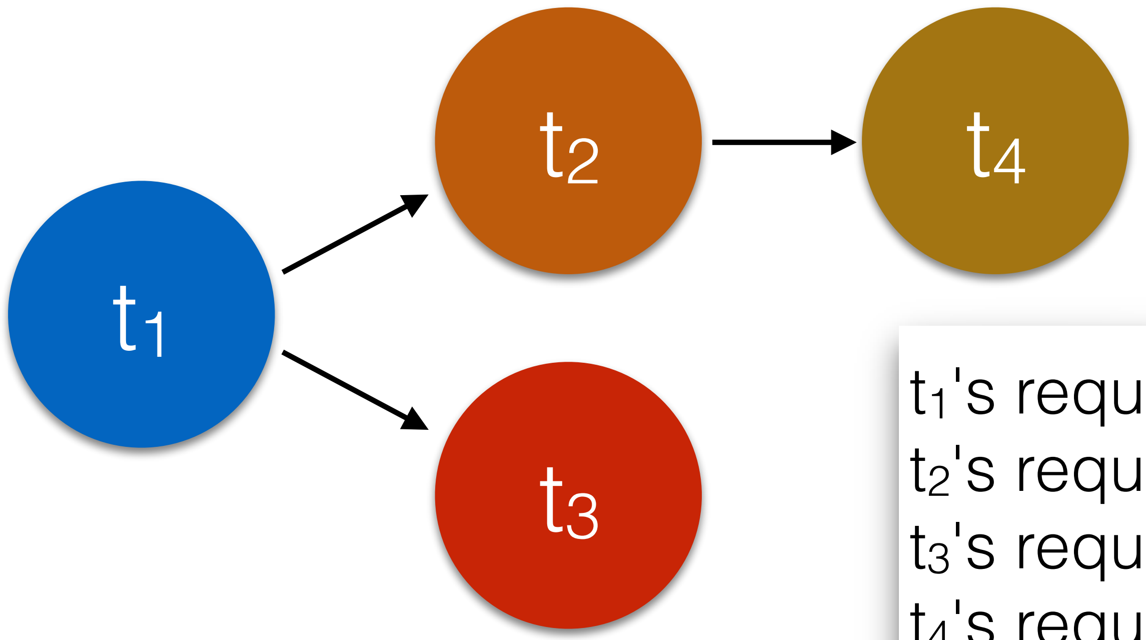
x’	t_1	t_2	t_3	t_4
e_1	0.5	0.5	0.5	0.5

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}



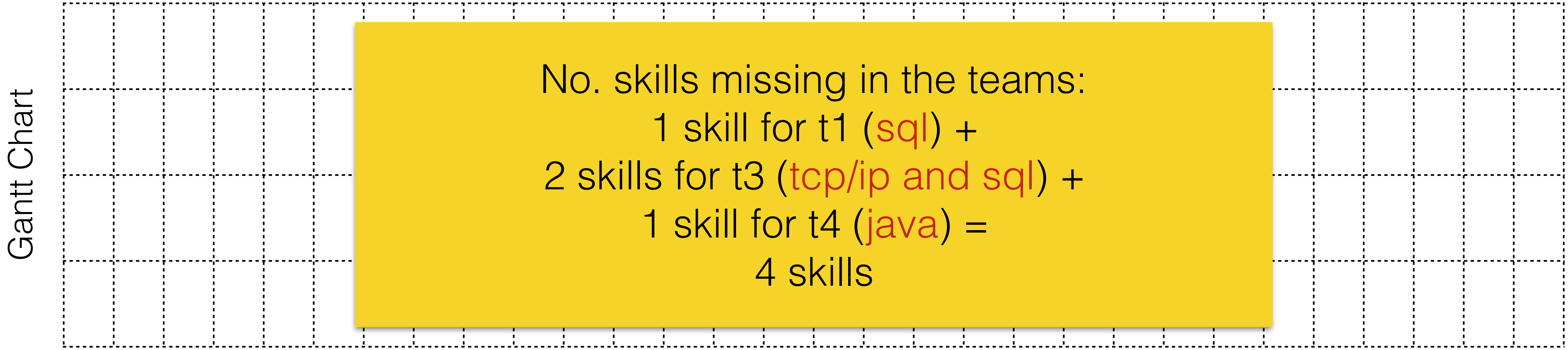
No. skills missing in the teams:
1 skill for t1 (sql) +
2 skills for t3 (tcp/ip and sql)=
3 skills

Example of Infeasible Schedule — Missing Skills



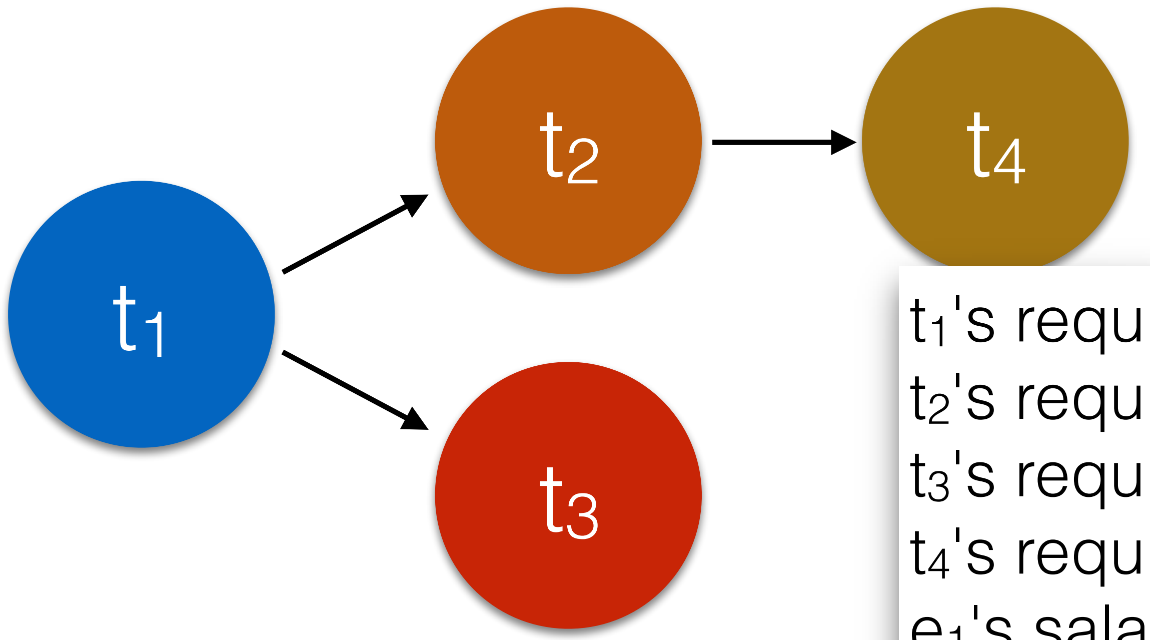
x'	t_1	t_2	t_3	t_4
e_1	0.5	0.5	0.5	0

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}



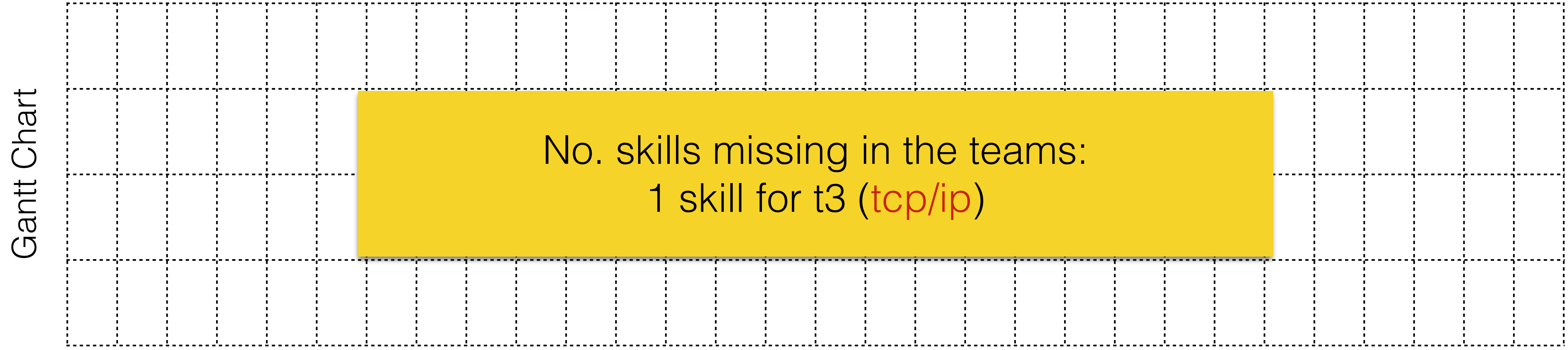
No. skills missing in the teams:
1 skill for t1 (sql) +
2 skills for t3 (tcp/ip and sql) +
1 skill for t4 (java) =
4 skills

Example of Infeasible Schedule — Missing Skills



x’	t_1	t_2	t_3	t_4
e_1	0.5	0.5	0.5	0.5
e_2	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}
e_2’s salary and skills: $1500 per full time month, {java,sql}



No. skills missing in the teams:
1 skill for t3 (tcp/ip)

Skills Constraint Formulation

Teams assigned to tasks must have all required skills to perform those tasks:

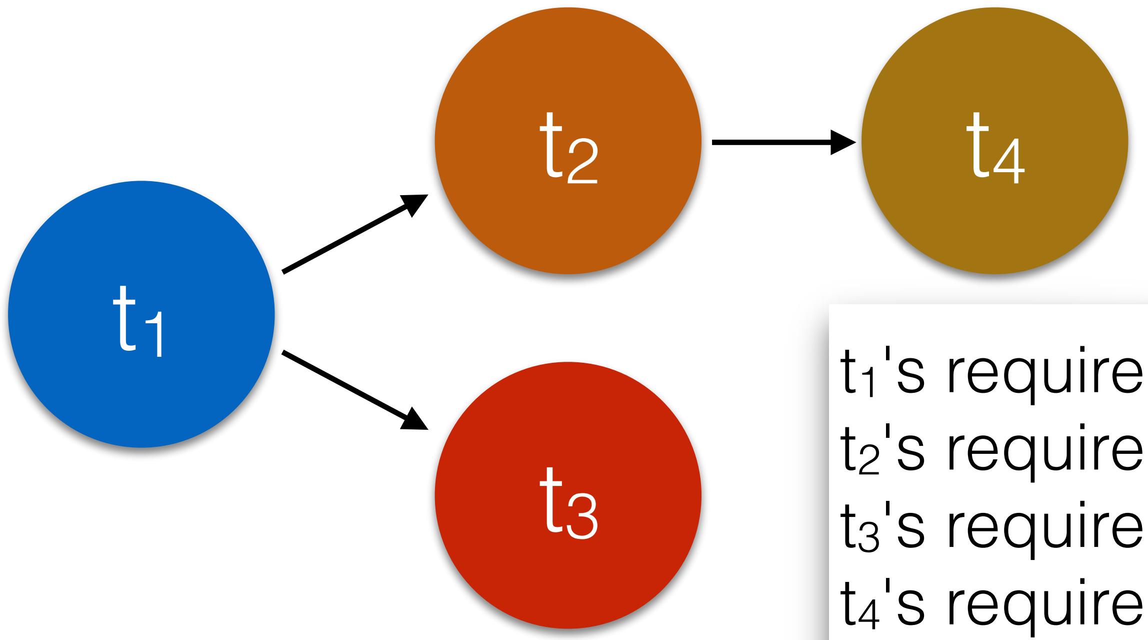
- numMissingSkills(**x’**) = 0.

```
integer numMissingSkills(x’)
missingSkills <— 0
For each task $j$ in {$t1, . . . , tm$}
    teamSkills <— set of skills of all employees i with $x'_{i,j} > 0$
    For each skill $s$ in $reqSk_j$
        If $s$ is not in $teamSkills$
            missingSkills <— missingSkills + 1
return missingSkills
```

Overwork Constraint

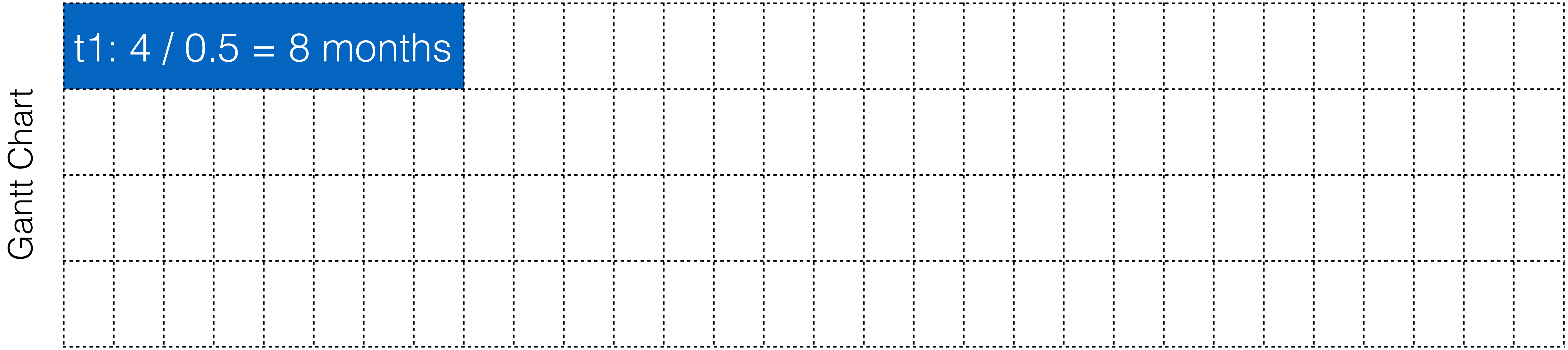
- No employee should exceed his/her maximum dedication during any period of the project.
- Consider that the maximum dedication is 1 for all employees.

Example of Infeasible Schedule — Overwork

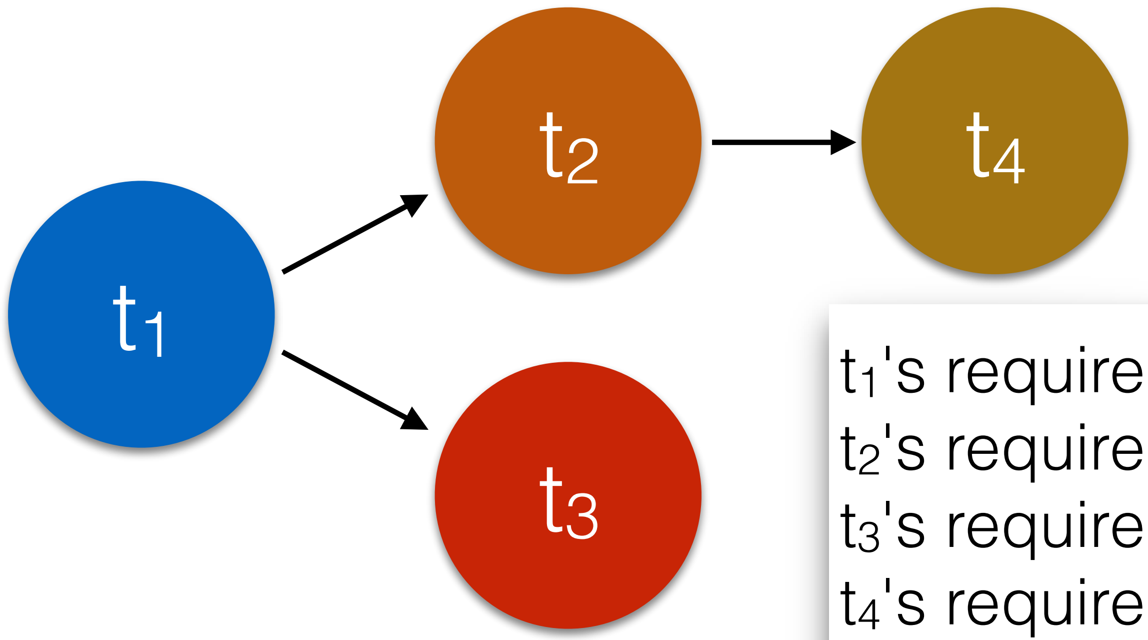


x’	t_1	t_2	t_3	t_4
e_1	0.5	1	0.5	0.5

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}

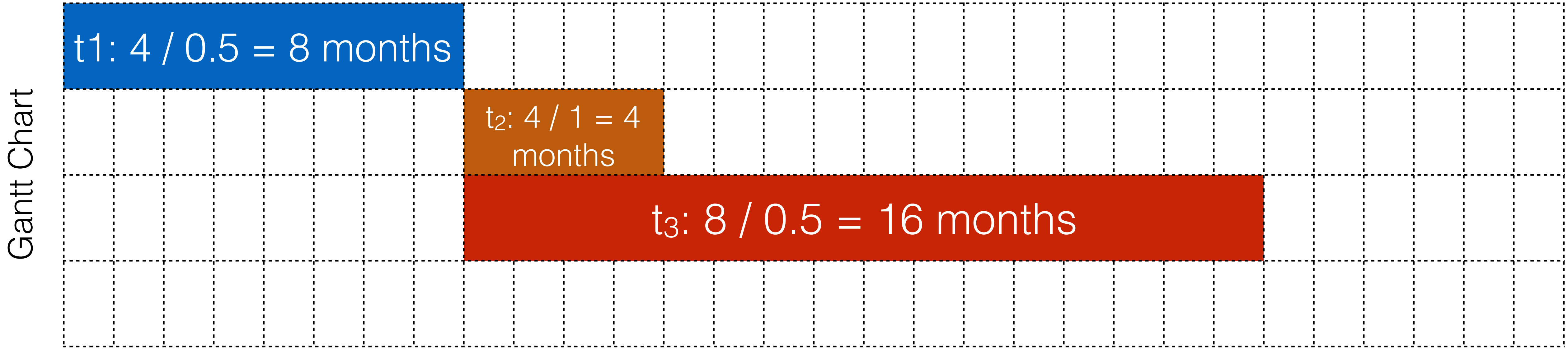


Example of Infeasible Schedule — Overwork

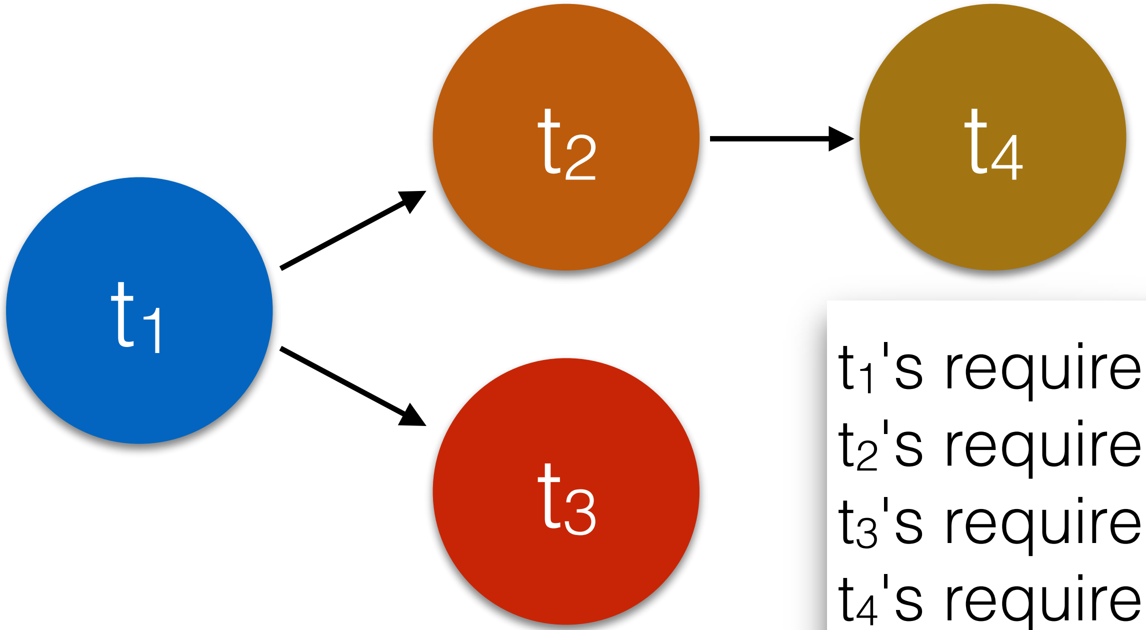


x’	t_1	t_2	t_3	t_4
e_1	0.5	1	0.5	0.5

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}



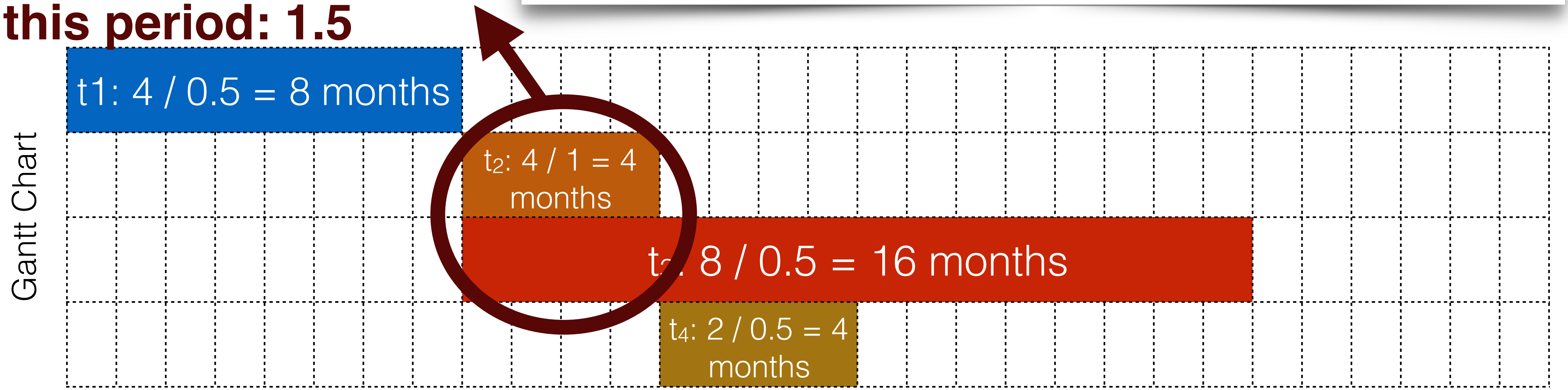
Example of Infeasible Schedule — Overwork



x’	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}

Dedication during this period: 1.5



Overwork Constraint

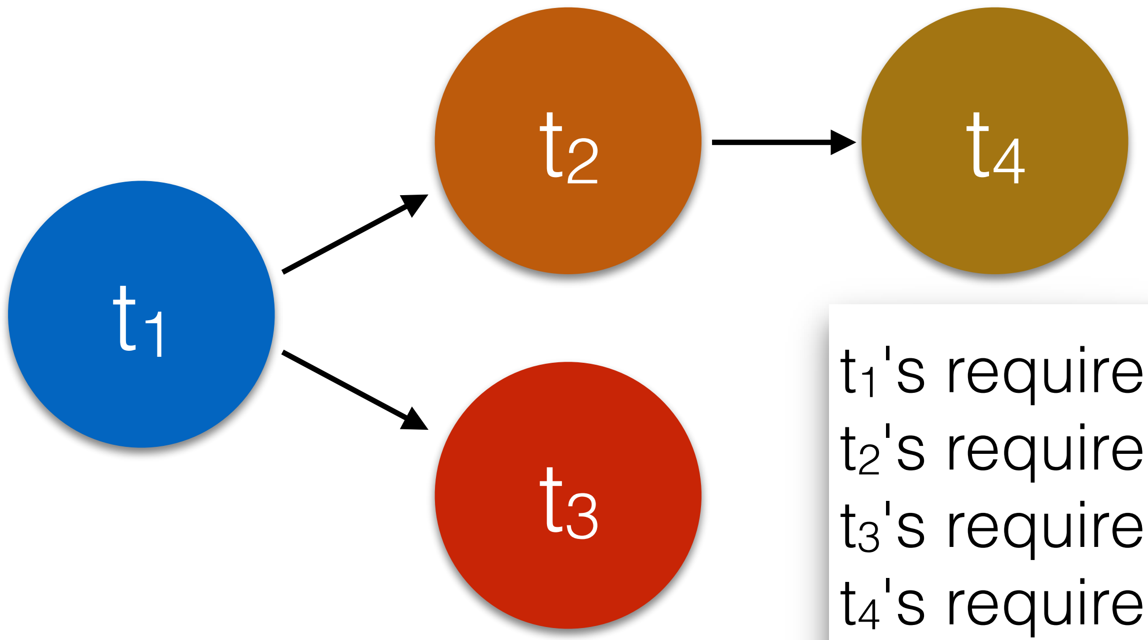
[Video from VideoVat.com posted by Marc RaZZ at: https://youtu.be/wXaPB7sIzNE]

Overwork Constraint Formulation

For any employee e_i and unit of time τ, the total dedication of e_i to tasks that are active at time τ should be at most 1:

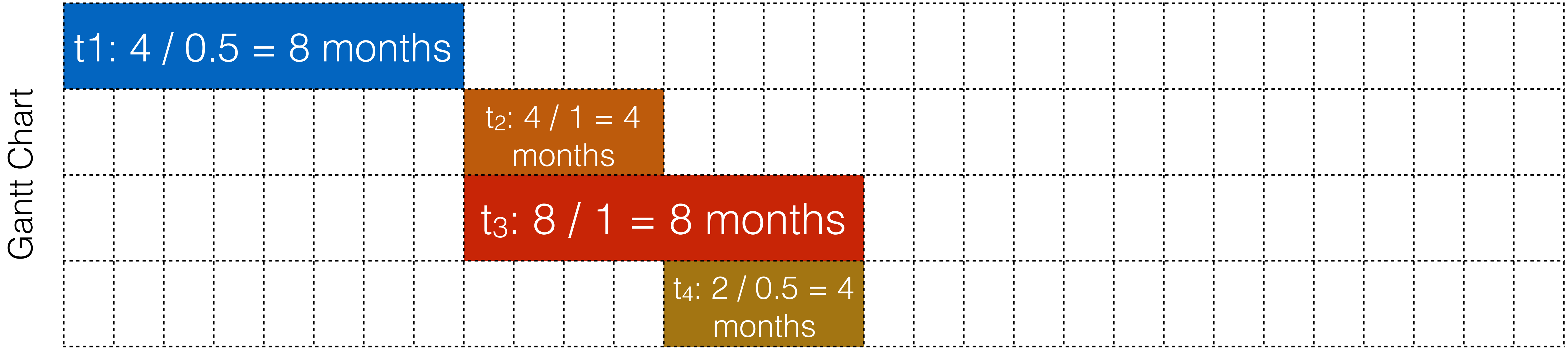
$$\sum_{t_j \text{ active at } \tau} x'_{ij} \leq 1$$

Is there overwork in the schedule below? How much?

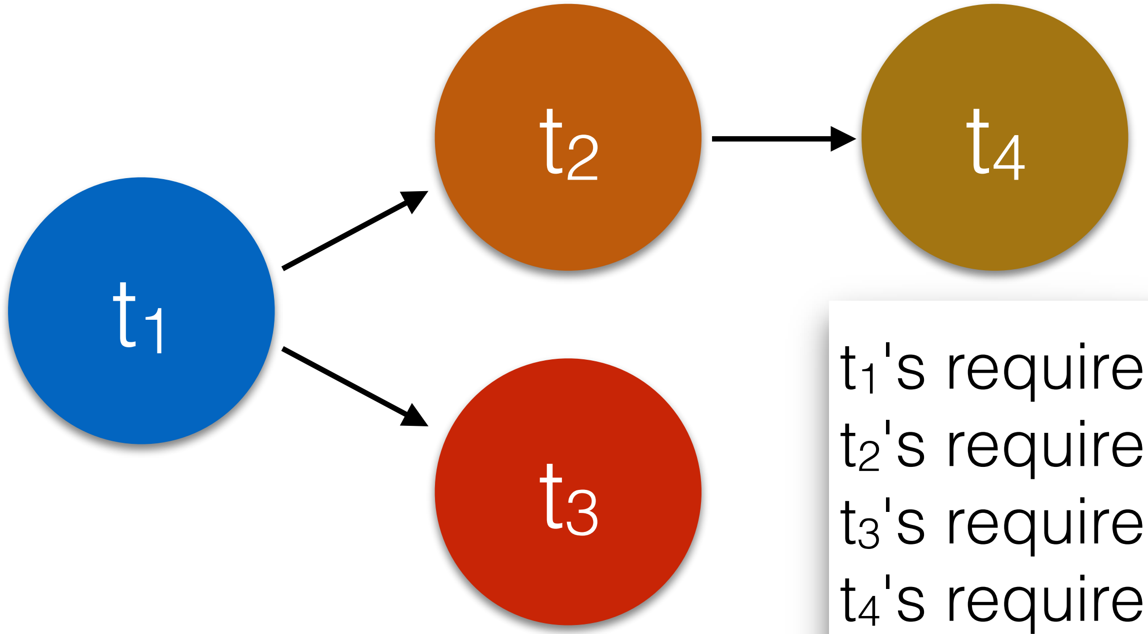


x’	t_1	t_2	t_3	t_4
e_1	0.5	1	1	0
e_2	0	0	0	0.5

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}



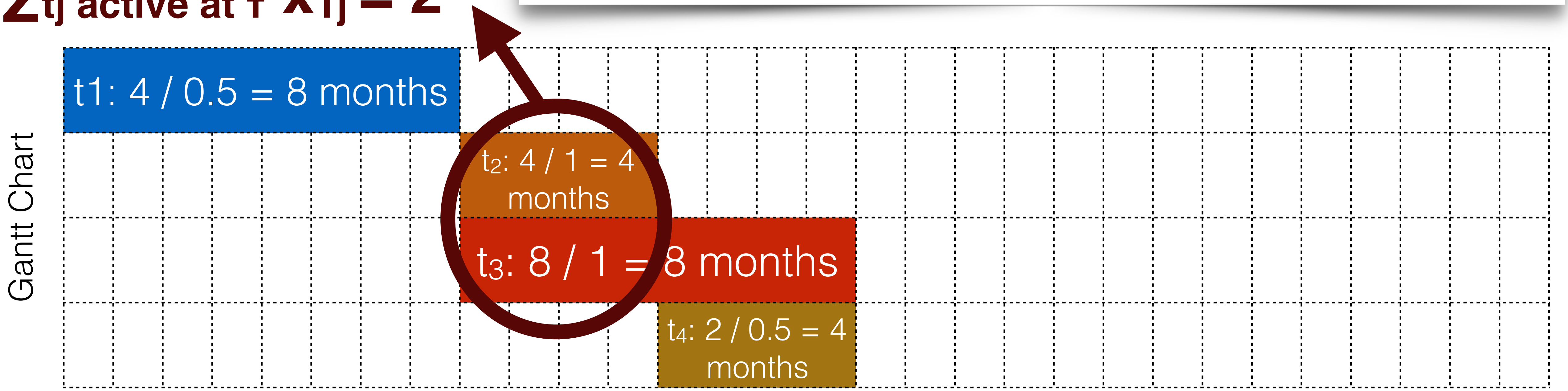
Is there overwork in the schedule below? How much?



x'	t_1	t_2	t_3	t_4
e_1	0.5	1	1	0
e_2	0	0	0	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}

$\Sigma_{t_j \text{ active at } \tau}\, x_{1j} = 2$



Dealing with Constraints

- Option 1: death penalty.
  - Problem: all infeasible solutions are equally bad, i.e., there is no guidance towards feasibility.
- Option 2: penalty based on level of infeasibility.
  - Ok for dealing with missing skills — the smaller the number of missing skills, the less infeasible the solution is.
  - Problem: overwork requires the right balance in the dedication of employees to tasks, being difficult to satisfy based on penalty functions.

Dealing with Constraints

- Option 3:
  - Missing skills: penalty based on level of infeasibility.
  - Overwork: normalisation decoder.

Missing Skills — Penalty Based on Level of Infeasibility

- Fitness function: $\text{fitness}(\mathbf{x'}) = w_{cost} * \text{cost}(\mathbf{x'}) + w_{dur} * \text{duration}(\mathbf{x'})$ (to be minimised)

- We can look at the two different components of the fitness function separately.

- Penalty:

Very large positive constant making the **cost** of infeasible solutions worse than that of any feasible ones

$$\text{cost}(\mathbf{x'}) = n_{cost\_penal} * \text{numMissingSkills}(\mathbf{x'})^2$$

$$\text{duration}(\mathbf{x'}) = n_{dur\_penal} * \text{numMissingSkills}(\mathbf{x'})^2$$

Very large positive constant making the **duration** of infeasible solutions worse than that of any feasible ones

Number of skills missing in the teams (squared)

Overwork — Normalisation Decoder

- Normalise dedications to deal with overwork so that total dedication is at most 1 when creating the gantt chart.

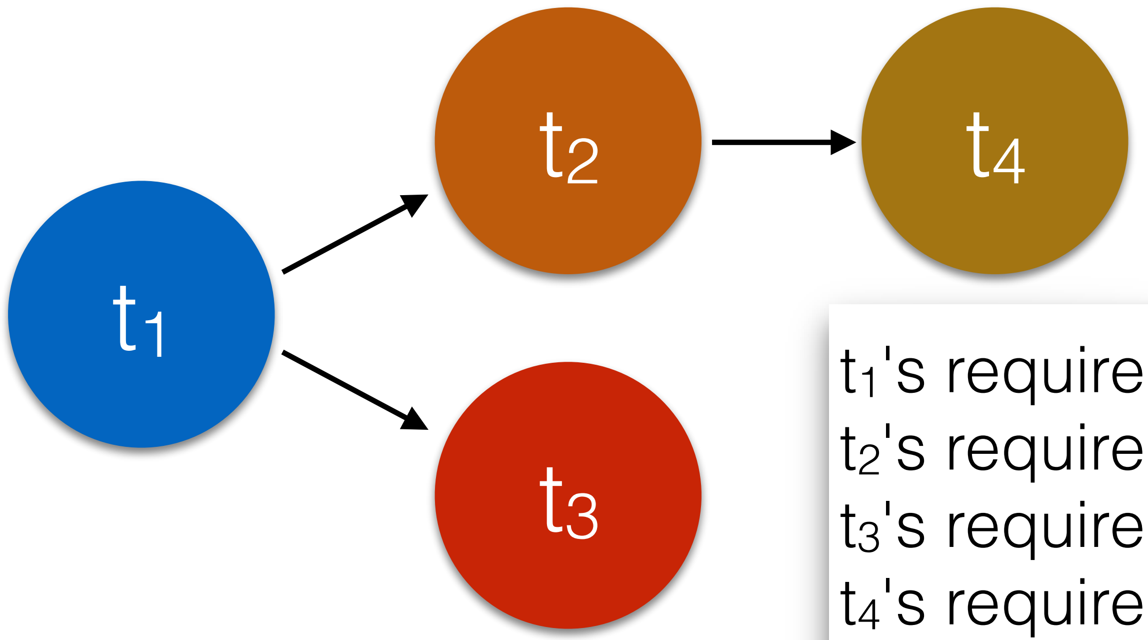
If employee e_i has overwork at any moment τ

$$d_{ij}(\tau) = x'_{ij} / \sum_{t_j \text{ active at } \tau} x'_{ij}$$

else

$$d_{ij}(\tau) = x'_{ij}$$

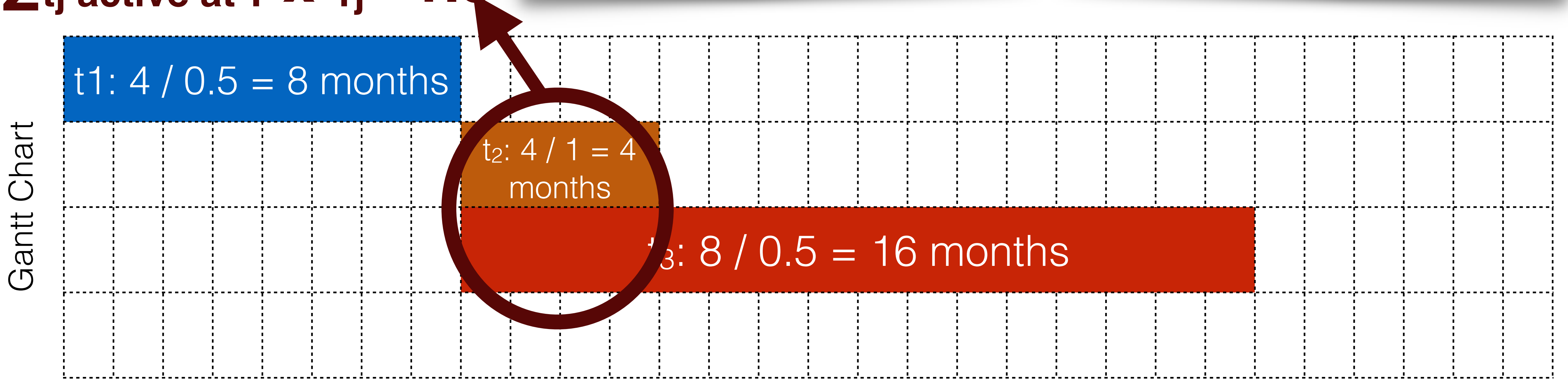
Example of Normalisation for Dealing with Overwork



x’	t_1	t_2	t_3	t_4
e_1	0.5	1	0.5	0.5

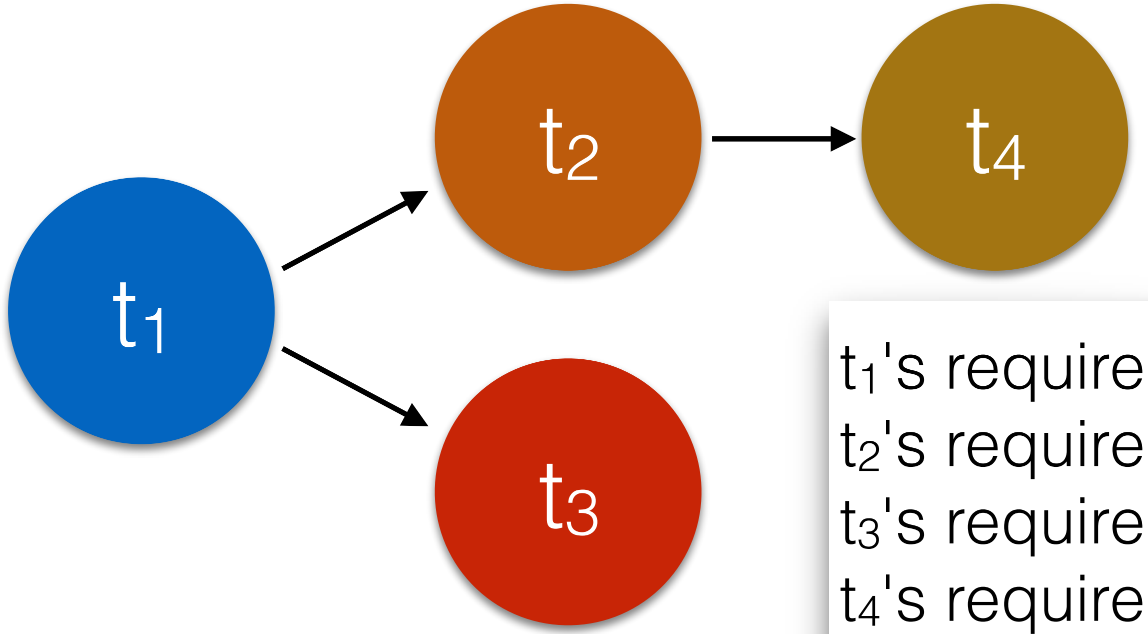
t_1's required effort and skills: 4 p-month, {sql, java}
t_2's required effort and skills: 4 p-month, {java}
t_3's required effort and skills: 8 p-month, {java}
t_4's required effort and skills: 2 p-month, {java}
e_1's salary and skills: $1000 per full time month, {sql,java}

$\Sigma_{tj \text{ active at } \tau}\ x'_{1j} = 1.5$



$$d_{ij}(\tau) = x'_{ij} \,/\, \Sigma_{tj \text{ active at } \tau}\ x'_{ij}$$

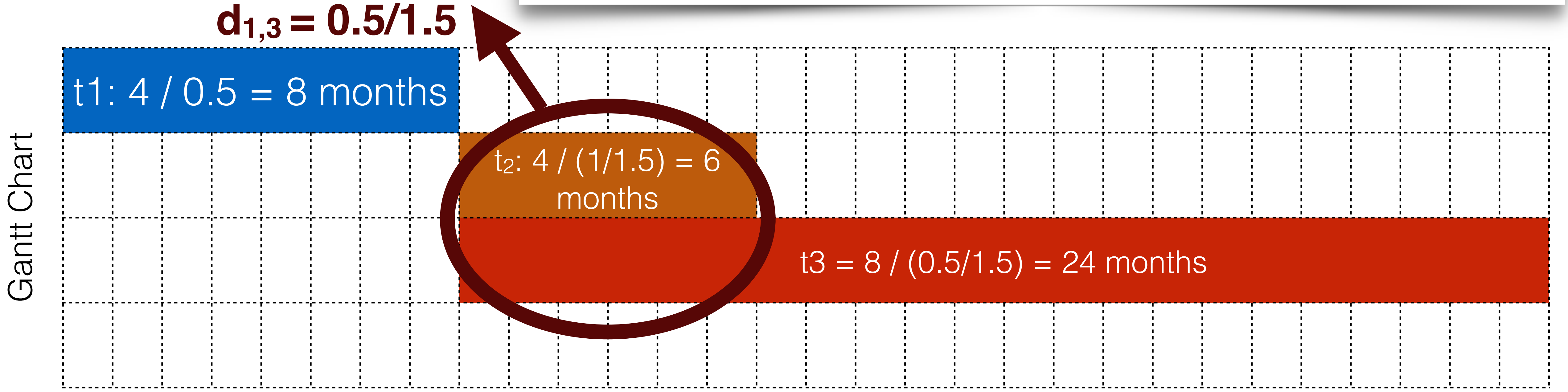
Example of Normalisation for Dealing with Overwork



x’	t_1	t_2	t_3	t_4
e_1	0.5	1	0.5	0.5

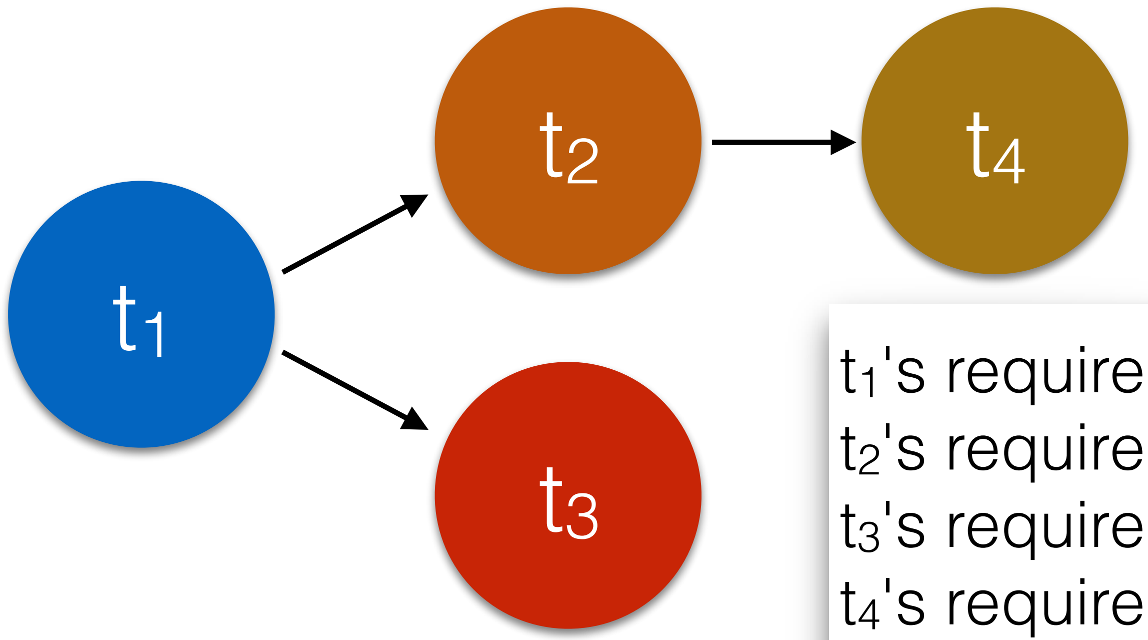
t_1's required effort and skills: 4 p-month, {sql, java}
t_2's required effort and skills: 4 p-month, {java}
t_3's required effort and skills: 8 p-month, {java}
t_4's required effort and skills: 2 p-month, {java}
e_1's salary and skills: \$1000 per full time month, {sql,java}

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



$$d_{ij}(\tau) = x'_{ij} / \sum_{t_j \text{ active at } \tau} x'_{ij}$$

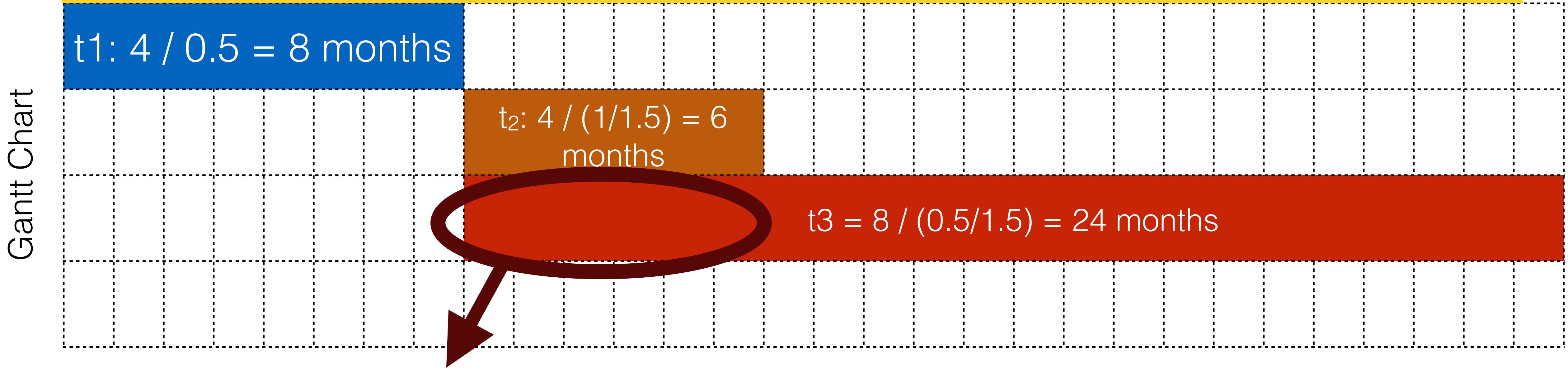
Example of Normalisation for Dealing with Overwork



x’	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}

How many person-months have been completed in the first 6 months of t3?



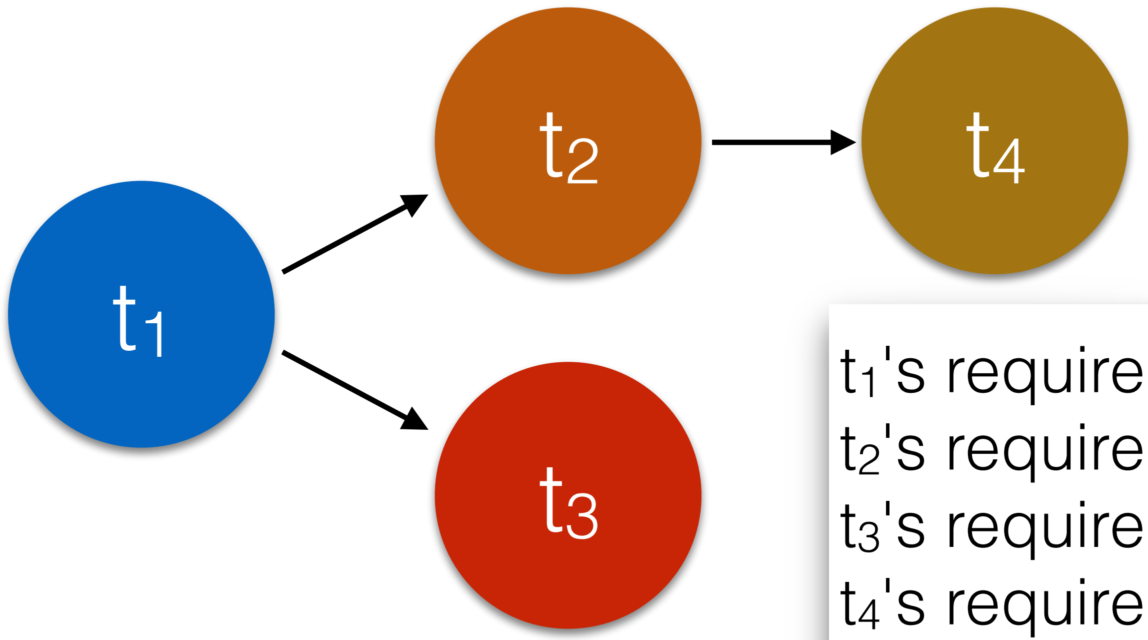
t1: 4 / 0.5 = 8 months

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

t3 = 8 / (0.5/1.5) = 24 months

6 months using dedication 0.5/1.5 —> 6 * 0.5/1.5 p-months = 2 p-months

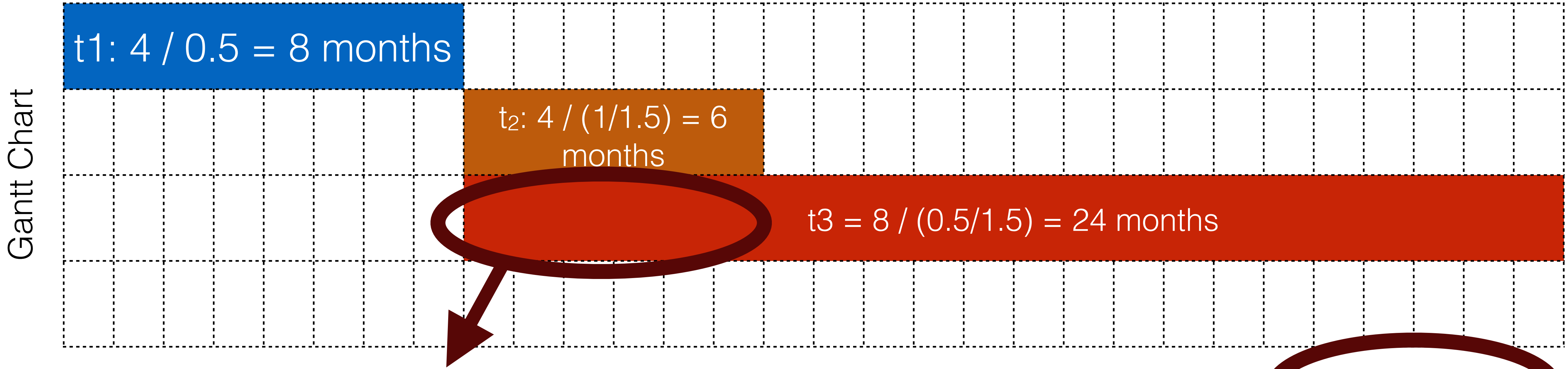
Example of Normalisation for Dealing with Overwork



x’	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}

How many p-months have to be completed after those 6 months of t3? 8 - 2 = 6 p-months



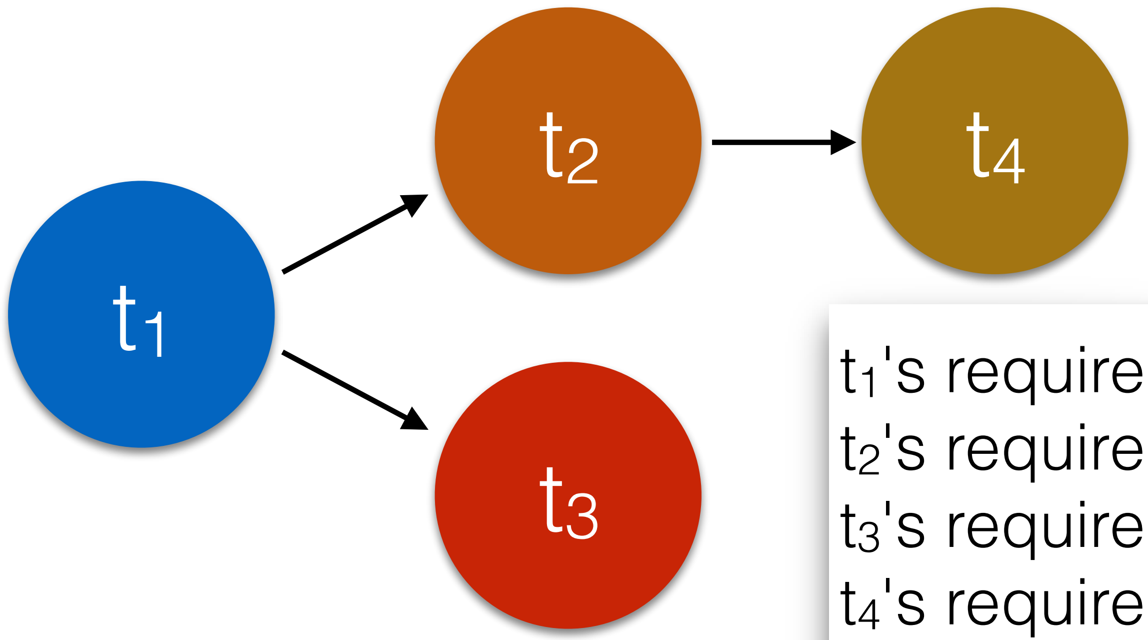
t1: 4 / 0.5 = 8 months

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

t3 = 8 / (0.5/1.5) = 24 months

6 months using dedication 0.5/1.5 —> 6 * 0.5/1.5 p-months = 2 p-months

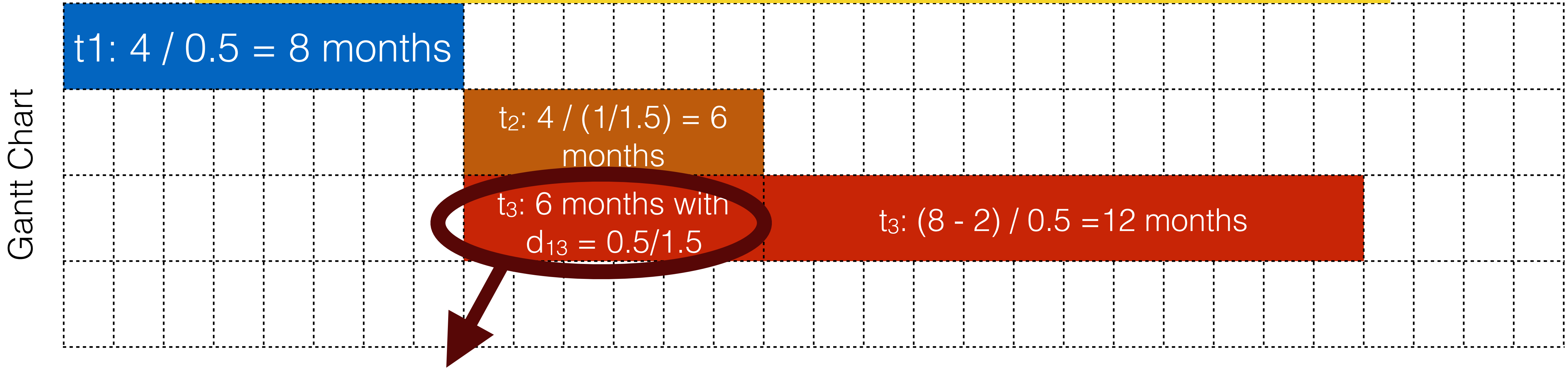
Example of Normalisation for Dealing with Overwork



x'	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}

How many months does it take to complete 6 p-months of t3?


Gantt Chart

t1: 4 / 0.5 = 8 months

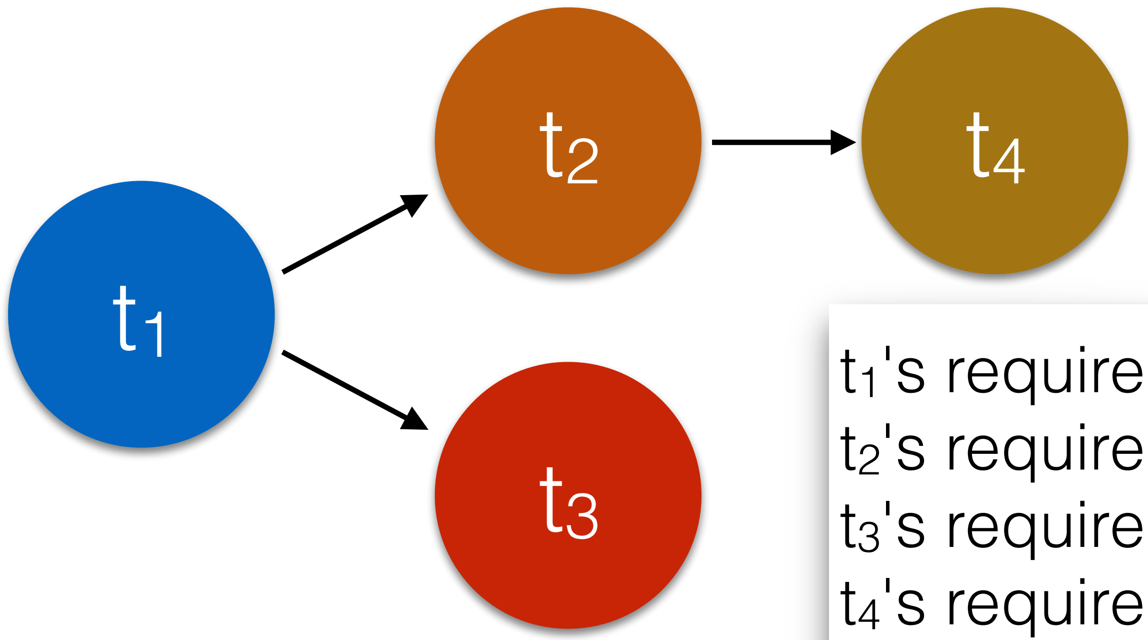
t_2: 4 / (1/1.5) = 6 months

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

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

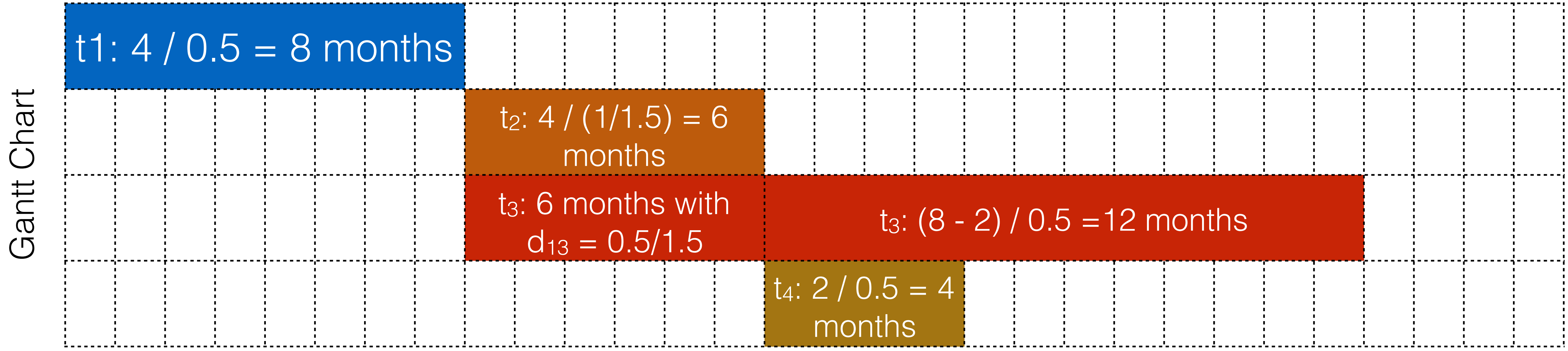
6 months using dedication 0.5/1.5 —> 6 * 0.5/1.5 p-months = 2 p-months

Example of Normalisation for Dealing with Overwork



x'	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}



Summary of Problem Formulation

- Design variable: matrix of dedications of employees to tasks.
- Objectives: cost and duration (to be minimised).
- Constraints: missing skills and overwork.

Summary of EA Design

- Representation: integer matrix of employees by tasks.
- Fitness function: fitness($\mathbf{x}'$) = w_{cost} * cost($\mathbf{x}'$) + w_{dur} * duration($\mathbf{x}'$)
  (to minimise)
- Dealing with constraints:
  - Overwork: normalisation decoder.
  - Skills missing: penalty based on infeasibility.
    cost($\mathbf{x}'$) = n_{cost_penal} * numMissingSkills($\mathbf{x}'$)^2
    duration($\mathbf{x}'$) = n_{dur_penal}* numMissingSkills($\mathbf{x}'$)^2
- Mutation: picks new dedication uniformly at random.
- Crossover: exchanges rows or columns.
- Parents selection: 2-Tournament selection.
- Survival selection: fitness-based delete-worst.

Termination condition: maximum number of generations

Summary

- What Software Project Scheduling Problem (SPSP) is.
- Why are automated optimisation methods are important for the SPSP.
- SPSP formulation as an optimisation problem.
- Suitable evolutionary algorithm for the SPSP.
- Dealing with constraints is a key issue for the SPSP.

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. Read all sections except for sections 6 and 8.

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6648326&tag=1