Génie logiciel pour la conception d'un Système d'Information

CSC4521

Voie d'Approfondissement

Intégration et Déploiement de Systèmes d'Information (VAP DSI)

Planning Tasks

paul.gibson@telecom-sudparis.eu

http://jpaulgibson.synology.me/~jpaulgibson/TSP/Teaching/CSC4521/CSC4521-PlanningTasks.pdf

Task Graphs: a simple yet powerful tool

Problem *Structure* <--……--> Solution *Structure*

Large gap => try an intermediate step

Problem ---> Task Graph ---> Solution

How to:

Problem -> Task Graph

split problem into tasks

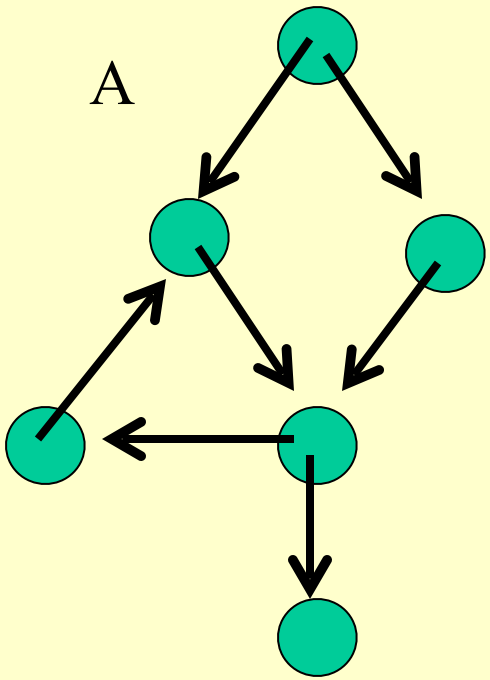
Task Graph -> Solution

map tasks to **parallel resources**

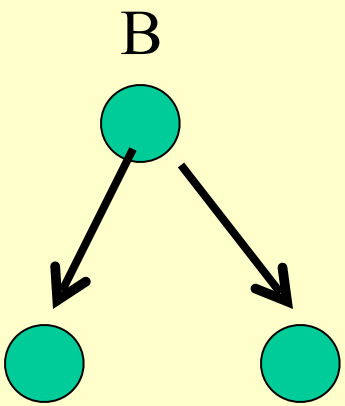
What Is A Task Graph?

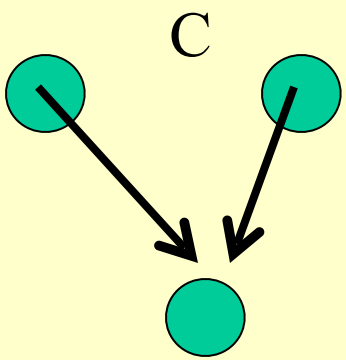
A *task graph* is a graph which has:

1 *root*, 1 *leaf*, no *cycles* and all *nodes connected*



A,B and C are graphs but they are not *task graphs*





Why are task graphs useful?

They help to identify an important property of the problem:

task dependency

They provide a formal model for scheduling which is amenable to:

rigorous mathematical analysis

They are simple, yet very powerful because they can be communicated to clients, managers and engineers:

non-ambiguous common language

There are standard extensions to the model which guard the simplicity and intuitiveness, but also enrich the semantics

HOW: Problem ---> Task Graph

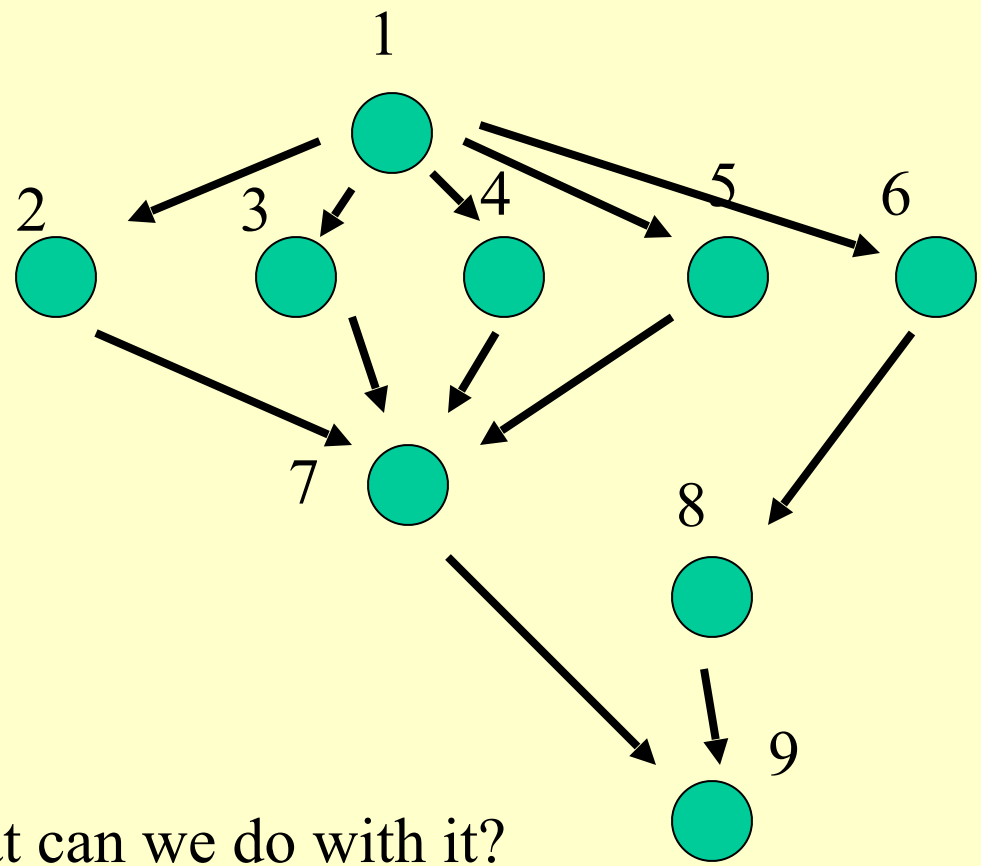
Task graphs are useful, but how do we create them?

There is a standard, informal, algorithm:

- Divide problem into set of n tasks
- Every task becomes a node in the task graph
- If task(x) cannot start before task(y) has finished then draw a line from node(y) to node(x)
- Identify (or create) starting and finishing tasks

The process (execution) flows through the task graph almost like *pipelining* in a single processor system

A Typical Task Graph



Question --- what can we do with it?

Answer --- we can construct *task sequences*

Task Sequences

A *task sequence* for a task graph, TG say,shows all *valid schedules* of the problem

ts =t1,…,tn is a valid task sequence for TG iff

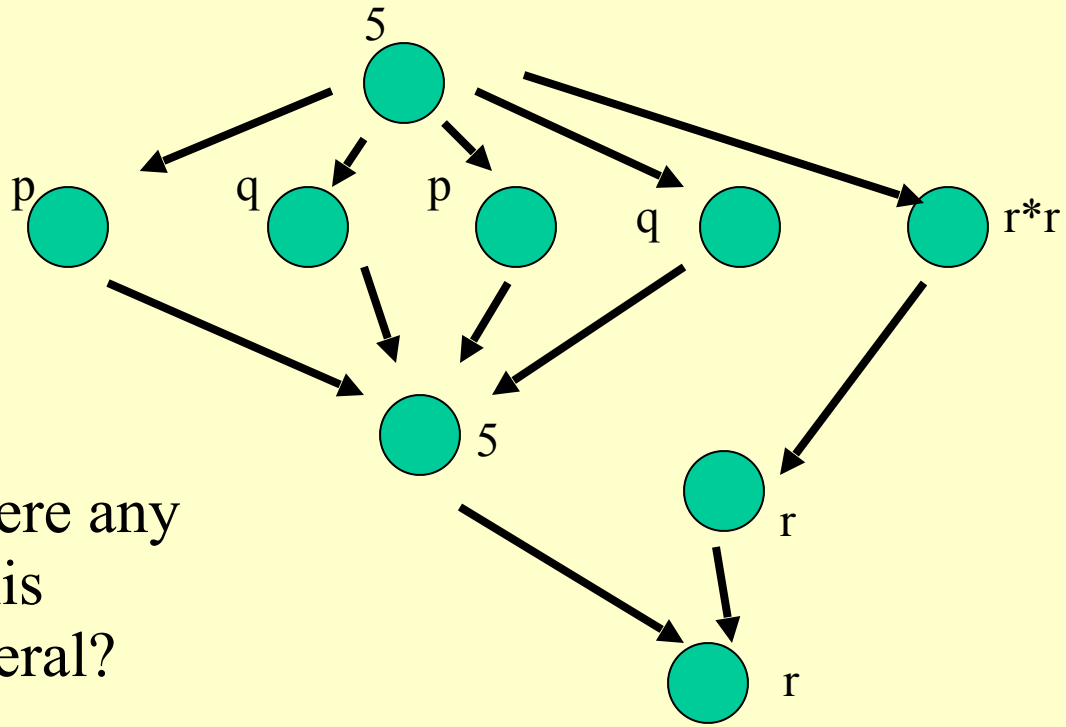
- t1 is a root node
- tn is a final (leaf) node
- there is a *1-1 and onto* mapping (isomorphism) between the tasks and the nodes in TG
- for all pairs of tasks ti,ti+1 in the sequence, there is no path from ti+1 to ti in TG

Annotated Task Graphs

In an annotated task graph:

For each task we annotate the TG with a value corresponding to *'task time'*

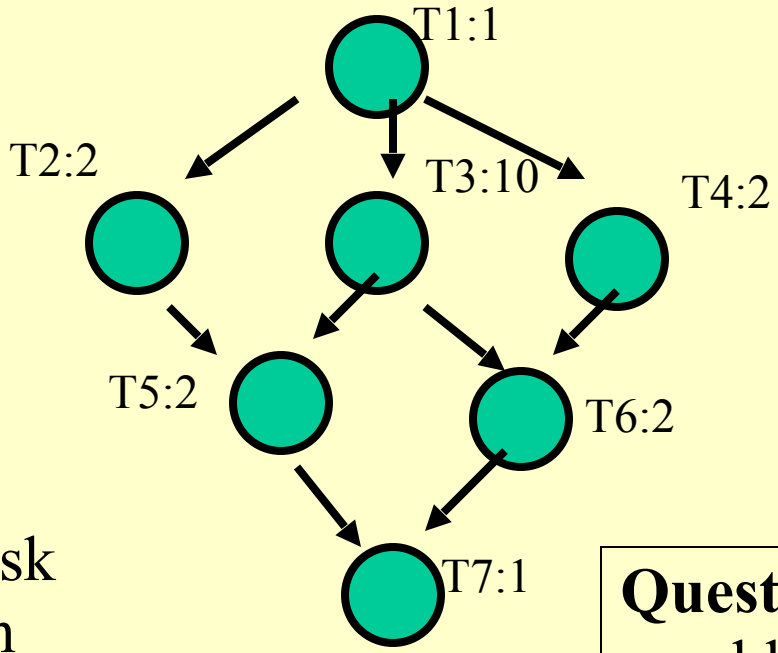
For example, consider a possible annotation for our previous graph ---



Question: are there any problems with this approach, in general?

Mapping task graphs to parallel resources

Example:



With 1 person, any task sequence will have an *execution time* = 1+2+10+2+2+2+1 = 20

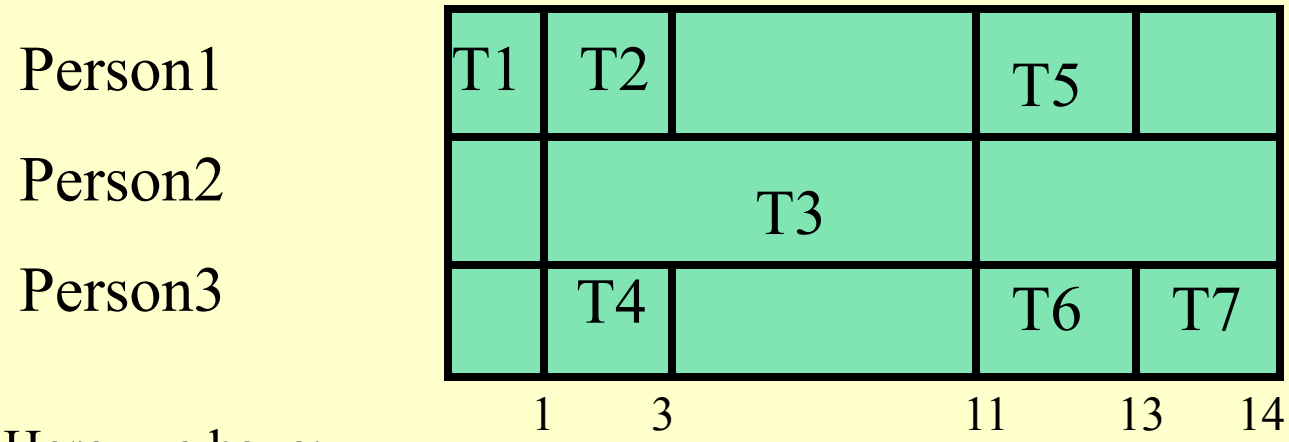
Goal: using people in parallel, reduce execution time

Question: what would our best execution time be in a system with an *unbounded* number of people?

Gantt Charts: Another useful structure transformation tool

To map a TG onto a parallel schedule, we use a *Gantt Chart*

Example: our previous example with 3 people



Here, we have:

- *execution time* = 14,
- *speed up* = time for single person/ new *execution time* = 20/14 = 1.4
- *efficiency* = speed up / number of people = 1.4/3 = 0.5

A First Analysis

In the previous problem, we did our analysis on a bounded number of people … why? … and why did we chose 3?

Question: how well could we do with more than 3 people?

Question: how well could we do with only 2 people?

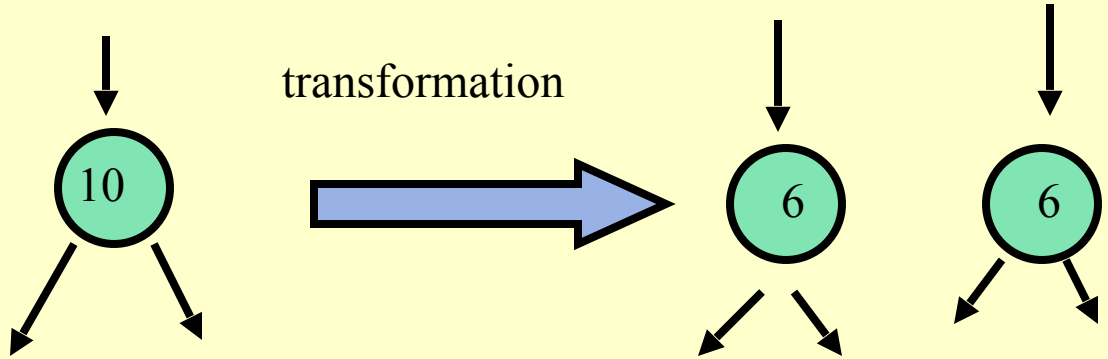
Question: why should we target our analysis on Task **T3**?

Potential improvement: split **T3** into subtasks and try again … this is known as a *task graph transformation*

Graph Transformation

Task Graph *Transformation* ---

T3 appears to be the *problem task* … what can we achieve if we divide it into 2 tasks. For example, two tasks taking *6 units* execution time each ---



Note: the transformation has cost us in terms of total work required --- it often costs more to split something up --- but the added structure means we can reduce execution time using parallel resources.

Question: after this transformation, can we do better with 3 people?

Extending the Task Graph Model

Annotated task graphs are good but they, like all models, abstract away from details. Sometimes these details are irrelevant to our analysis but other times they can have a large impact.

In such a situation, our only choice is to extend the model.

For example:

- different people may work at different speeds
- some people may be unreliable
- there may be communication delays between people/tasks
- there may be *fairness concerns*
- tasks may be people specific; people may be task specific
- there may be variable task times ...