Scheduling Techniques

Dra. Mª Dolores Rodríguez Moreno

Objectives

Specific Objectives

- To understand what scheduling is
- Main techniques

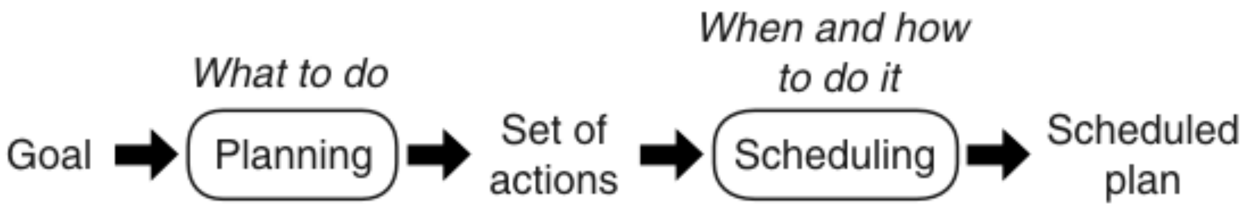
Source

- Stuart Russell & Peter Norvig (2009). Artificial Intelligence: A Modern Approach. (3rd Edition). Ed. Pearsons
- Dechter, Meiri, and Pearl (1991). Simple Temporal Networks. Artificial Intelligence 49:61–95.
- Dana Nau's slides for Automated Planning. Licensed under License https://creativecommons.org/licenses/by-nc-sa/2.0/

Outline

- **Introduction**
- Scheduling Problem (SP)
- Types of SP
- Techniques
    - Operational Research (OR)
    - Constraints Satisfaction Problems (CSPs)
    - Temporal Constraints Satisfaction Problems (TCSP)
- Conclusions

Introduction (I)



- Scheduling has usually been addressed separately from planning
- In some cases, cannot decompose planning and scheduling so cleanly
- There is not specific language

Introduction (II)

- Given:
  - actions to perform
  - set of resources to use
  - time constraints
- Objective:
  - allocate times and resources to the actions
- What is a resource?
  - Something needed to carry out the action
  - Usually represented as a numeric quantity
  - Actions modify it in a *relative* way
  - Several concurrent actions may use the same resource

Introduction (III)

- Scheduling Problem (SP)
  - Set of resources and their future availability
  - Actions and their resource requirements
  - Constraints
  - Cost function
- Output
  - Allocations of resources and start times to actions
  - Must meet the constraints and resource requirements

Outline

- Introduction
- **Scheduling Problem (SP)**
- Types of SP
- Techniques
  - Operational Research (OR)
  - Constraints Satisfaction Problems (CSPs)
  - Temporal Constraints Satisfaction Problems (TCSP)
- Conclusions

SP: Actions

- Action a
  - resource requirements: which resources, what quantities
  - usually, upper and lower bounds on start and end times
    - Start time $s(a) \in [s_{min}(a), s_{max}(a)]$
    - End time $e(a) \in [e_{min}(a), e_{max}(a)]$
- Non-preemptive action: cannot be interrupted
  - Duration $d(a) = e(a) - s(a)$
- Preemptive action: can interrupt and resume
  - Duration $d(a) = \sum_{i \in I} d_i(a) \leq e(a) - s(a)$
  - Can have constraints on the intervals

SP: Reusable Resources

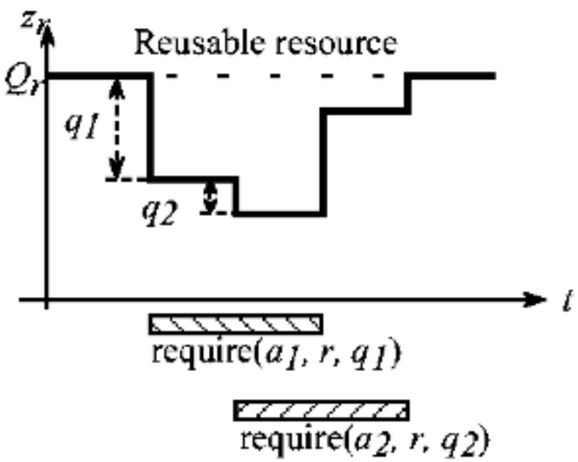
- A *reusable* resource is "borrowed" by an action, and released afterward
  - e.g., use a tool, return it when done
- Total capacity Q_i for r_i may be either discrete or continuous
  - Current level $z_i(t) \in [0, Q_i]$ is -- $z_i(t)$ = how much of r_i is currently available
- If action requires quantity q of resource r_i,
  - Then decrease z_i by q at time $s(a)$ and increase z_i by q at time $e(a)$
- Example: the two grippers in the robot problem
  - We might represent this as $Q_i = 2$
  - One of them in use at time t: $z_i(t) = 2 - 1 = 1$

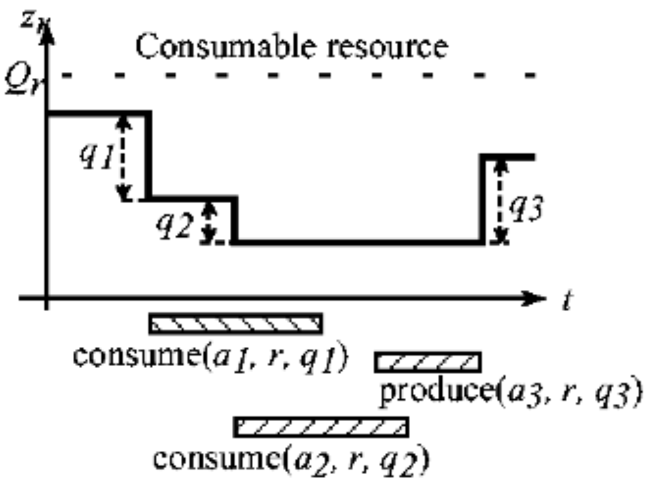
SP: Consumable Resources

- A *consumable* resource is used up (or in some cases produced) by an action (e.g., fuel)
- Like before, we have total capacity Q_i and current level $z_i(t)$
- If action requires quantity q of r_i
  - Decrease z_i by q at time $s(a)$
  - Don't increase z_i at time $e(a)$

SP: Resources

- An action's resource requirement is a conjunct of assertions
  - consume(a,r_j,q_j) & …
- or a disjunct if there are alternatives
  - consume(a,r_j,q_j) ∨ …
- z_i is called the "resource profile"





SP: Time constraints

- Bounds on start and end points of an action
  - absolute times
    - e.g., a deadline: $e(a) \leq u$
    - release date: $s(a) \geq v$
  - relative times
    - latency: $u \leq s(b) - e(a) \leq v$
    - total extent: $u \leq e(a) - s(a) \leq v$
- Constraints on availability of a resource
  - e.g., can only communicate with a satellite at certain times

SP: Costs

- may be fixed
- may be a function of quantity and duration
  - e.g., a set-up cost to begin some activity,
    plus a run-time cost that's proportional to the amount of time
- e.g., suppose *a* follows *b*
  - cost $c_r(a,b)$ for *a*
  - duration $d_r(a,b)$, i.e., $s(b) \geq e(a) + d_r(a,b)$
- Objective: minimize some function of the various costs and/or end-times
  - The makespan or maximum ending time
  - The total number of resources allocated …

Outline

- Introduction
- Scheduling Problem (SP)
- **Types of SP**
- Techniques
  - Operational Research (OR)
  - Constraints Satisfaction Problems (CSPs)
  - Temporal Constraints Satisfaction Problems (TCSP)
- Conclusions

Types of SP: machine scheduling

- Machine i: unit capacity (in use or not in use)
- Job j: partially ordered set of actions $a_{j1}, \ldots, a_{jk}$
- Schedule:
  - A machine i for each action a_{jk}
  - A time interval during which i processes a_{jk}
  - No two actions can use the same machine at once
- Actions in different jobs are completely independent
- Actions in the same job cannot overlap
  - e.g., Actions to be performed on the same physical object

Types of SP: single-stage machine scheduling

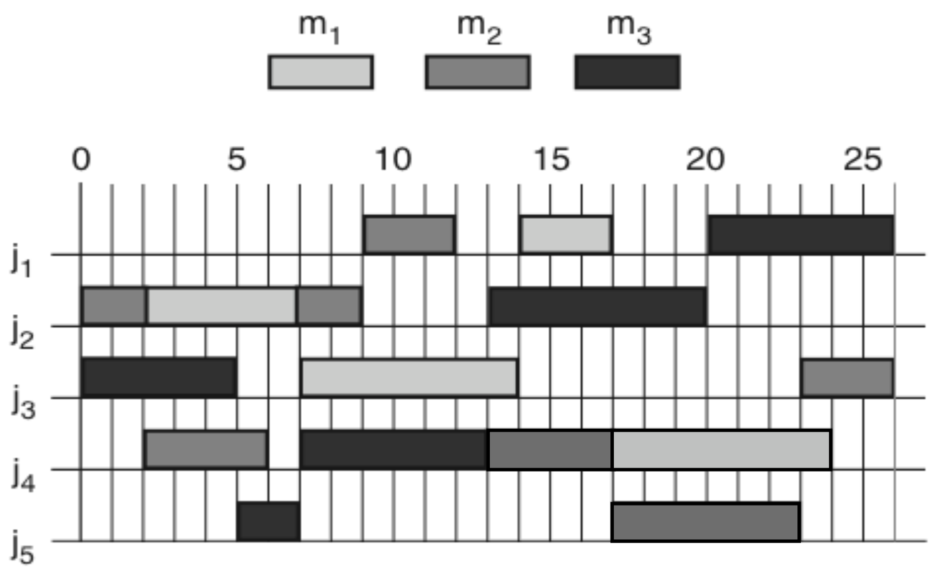
- Each job is a single action, and can be processed on any machine
- Identical parallel machines
  - Processing time p_j is the same regardless of which machine
  - Thus we can model all m machines as a single resource of capacity m
- Uniform parallel machines
  - Machine i has speed(i); time for j is p_j/speed(i)
- Unrelated parallel machines
  - Different time for each combination of job and machine

Types of SP: multiple-stage scheduling

- Job contains several actions
- Each requires a particular machine
- **Flow-shop** problems:
  - Each job j consists of exactly m actions $\{a_{j1}, a_{j2}, \ldots, a_{jm}\}$
  - Each a_{ji} needs to be done on machine i
  - Actions must be done in order $a_{j1}, a_{j2}, \ldots, a_{jm}$
- **Open-shop** problems
  - Like flow-shop, but the actions can be done in any order
- **Job-shop** problems (general case)
  - Constraints on the order of actions, and which machine for each action

Example: Job Shop

- Machines m_1, m_2, m_3 and jobs $j_1, \ldots, j_5$
- j_1: $\langle m_2(3), m_1(3), m_3(6) \rangle$
  - *i.e.*, m_2 for 3 time units
    then m_1 for 3 time units
    then m_3 for 6 time units
- j_2: $\langle m_2(2), m_1(5), m_2(2), m_3(7) \rangle$
- j_3: $\langle m_3(5), m_1(7), m_2(3) \rangle$
- j_4: $\langle m_2(4), m_3(6), m_2(4), m_1(7) \rangle$
- j_5: $\langle m_3(2), m_2(6) \rangle$



Outline

- Introduction
- Scheduling Problem (SP)
- Types of SP
- **Techniques**
  - **Operational Research (OR)**
  - Constraints Satisfaction Problems (CSPs)
  - Temporal Constraints Satisfaction Problems (TCSP)
- Conclusions

Operational Research

- Different approaches: Branch and bound, lineal programming, lagrangian relaxation…
- Integer Programming (IP) formulations
  - Set of constraints C, all are linear inequalities
  - Linear objective function f
  - Find a point $p=(x_1,\dots, x_n)$ such that
    - p satisfies C
    - p is integer-valued, i.e., every x_i is an integer
    - no other integer-valued point p' satisfies C and has $f(p') < f(p)$
- A huge number of problems can be translated into this format
- Several commercial IP solvers

Outline

- Introduction
- Scheduling Problem (SP)
- Types of SP
- Techniques
  - Operational Research (OR)
  - **Constraints Satisfaction Problems (CSPs)**
  - Temporal Constraints Satisfaction Problems (TCSP)
- Conclusions

Outline

- Introduction
- Scheduling Problem (SP)
- Types of SP
- Techniques
    - Operational Research (OR)
    - Constraints Satisfaction Problems (CSPs)
    - **Temporal Constraints Satisfaction Problems (TCSP)**
- Conclusions

Outline

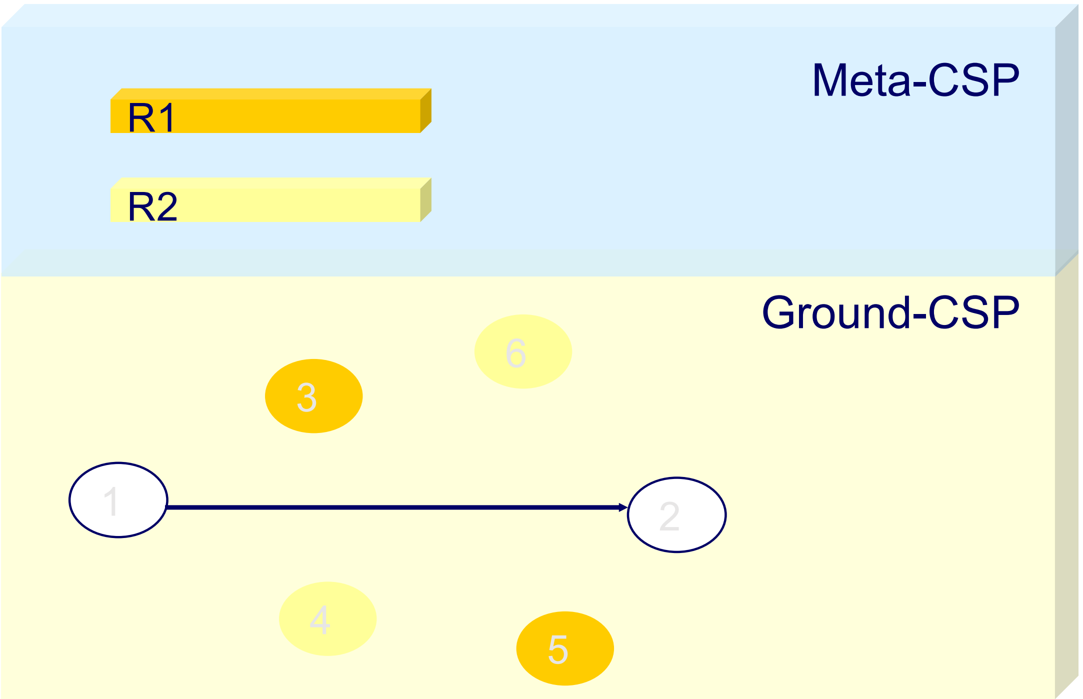
- Introduction
- Scheduling Problem (SP)
- Types of SP
- Techniques
  - Operational Research (OR)
  - Constraints Satisfaction Problems (CSPs)
  - Temporal Constraints Satisfaction Problems (TCSP)
- **Conclusions**

Conclusions

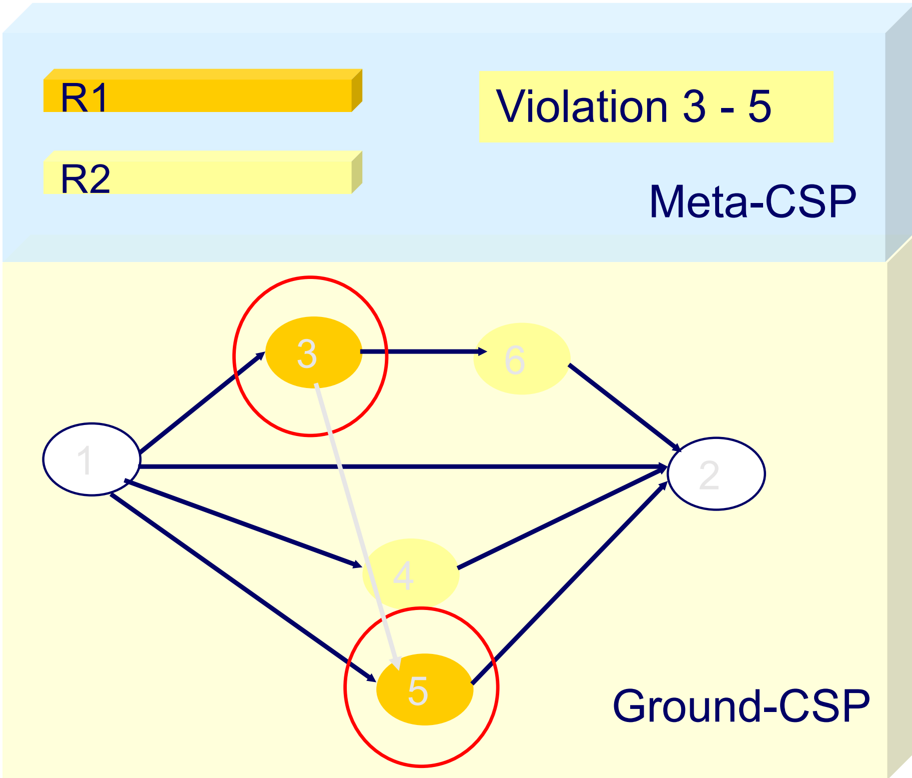
- SP techniques: OR & CSP
- For time, TCSP → uses STN
- How can we integrate P&S?

Integrating P&S

- CSP approach



Integrating P&S



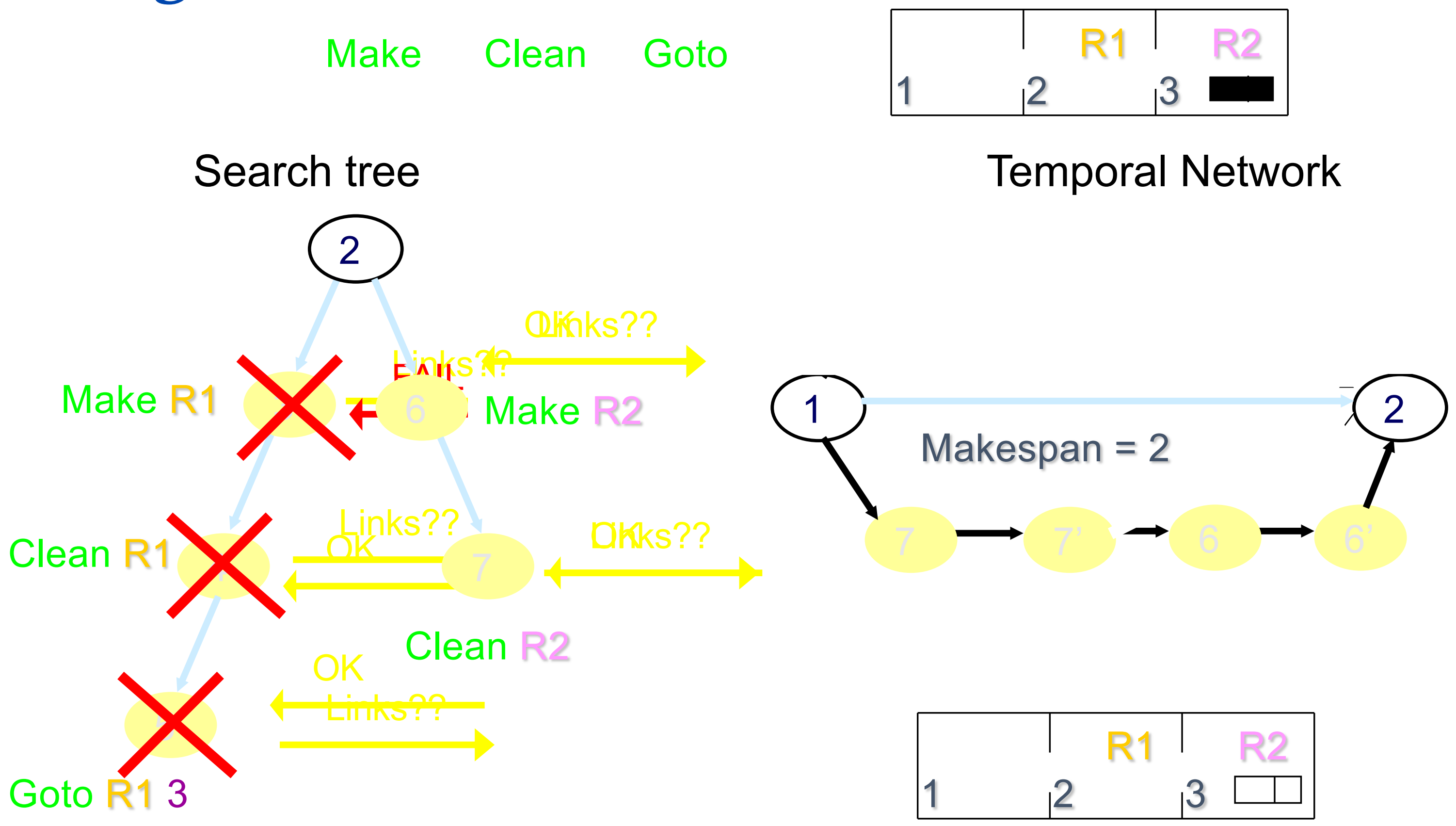
Propagation

Compute

Select

Resolve

Integrating P&S



Scheduling Techniques

Dra. Mª Dolores Rodríguez Moreno

Backtracking search

```
function BACKTRACKING-SEARCH( csp) returns a solution, or failure
   return RECURSIVE-BACKTRACKING({}, csp)

function RECURSIVE-BACKTRACKING( assignment,csp) returns a solution, or
failure
   if assignment is complete then return assignment
   var ← SELECT-UNASSIGNED-VARIABLE( Variables[csp], assignment, csp)
   for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
      if value is consistent with assignment according to Constraints[csp] then
         add { var = value } to assignment
         result ← RECURSIVE-BACKTRACKING(assignment, csp)
         if result ≠ failue then return result
         remove { var = value } from assignment
   return failure
```

Arc consistency AC-3

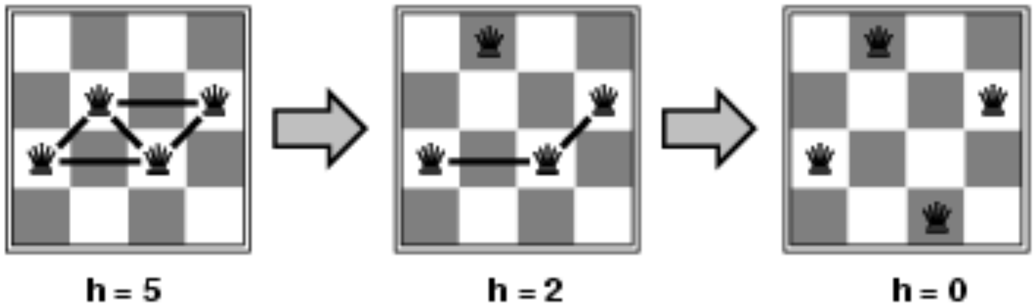
```
function AC-3( csp) returns the CSP, possibly with reduced domains
  inputs: csp, a binary CSP with variables {X1, X2, ..., Xn}
  local variables: queue, a queue of arcs, initially all the arcs in csp

  while queue is not empty do
    (Xi, Xj) ← REMOVE-FIRST(queue)
    if RM-INCONSISTENT-VALUES(Xi, Xj) then
      for each Xk in NEIGHBORS[Xi] do
        add (Xk, Xi) to queue
```

```
function RM-INCONSISTENT-VALUES( Xi, Xj) returns true iff remove a value
  removed ← false
  for each x in DOMAIN[Xi] do
    if no value y in DOMAIN[Xj] allows (x,y) to satisfy constraint(Xi, Xj)
      then delete x from DOMAIN[Xi]; removed ← true
  return removed
```

- Time complexity: O(n2d3)

Example: 4-Queens

- States: 4 queens in 4 columns ($4^4 = 256$ states)
- Actions: move queen in column
- Goal test: no attacks
- Evaluation: $h(n)$ = number of attacks



- Given random initial state, can solve n-queens in almost constant time for arbitrary n with high probability (e.g., n = 10,000,000)