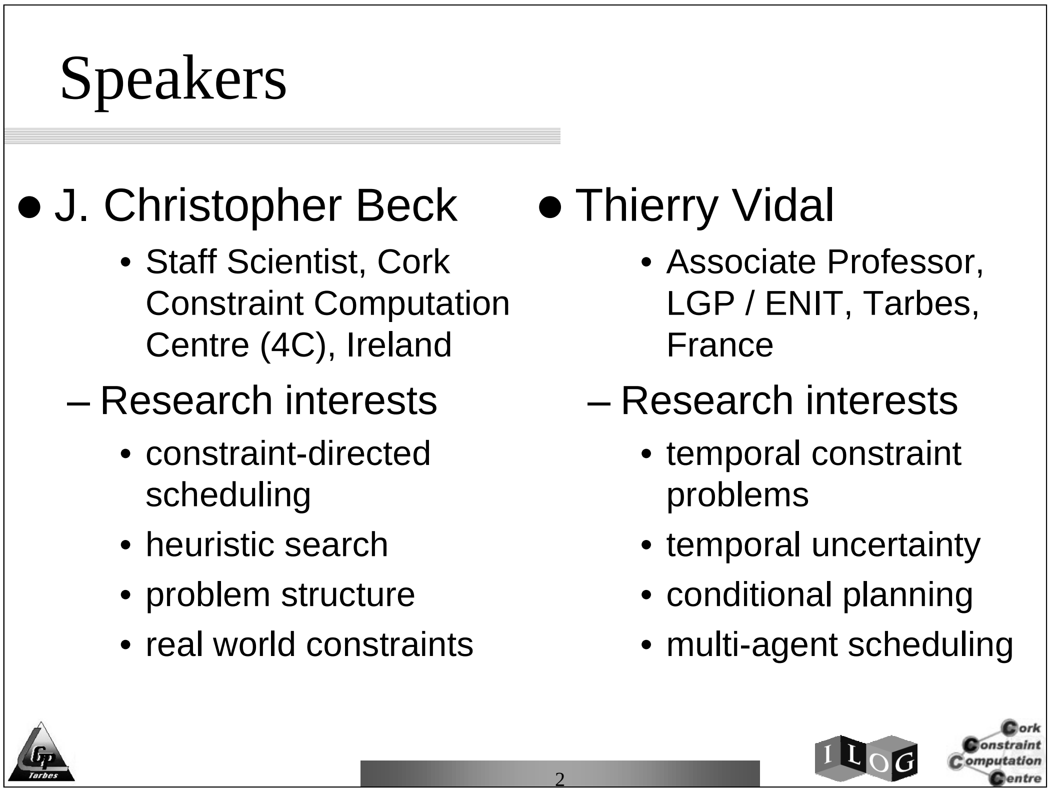
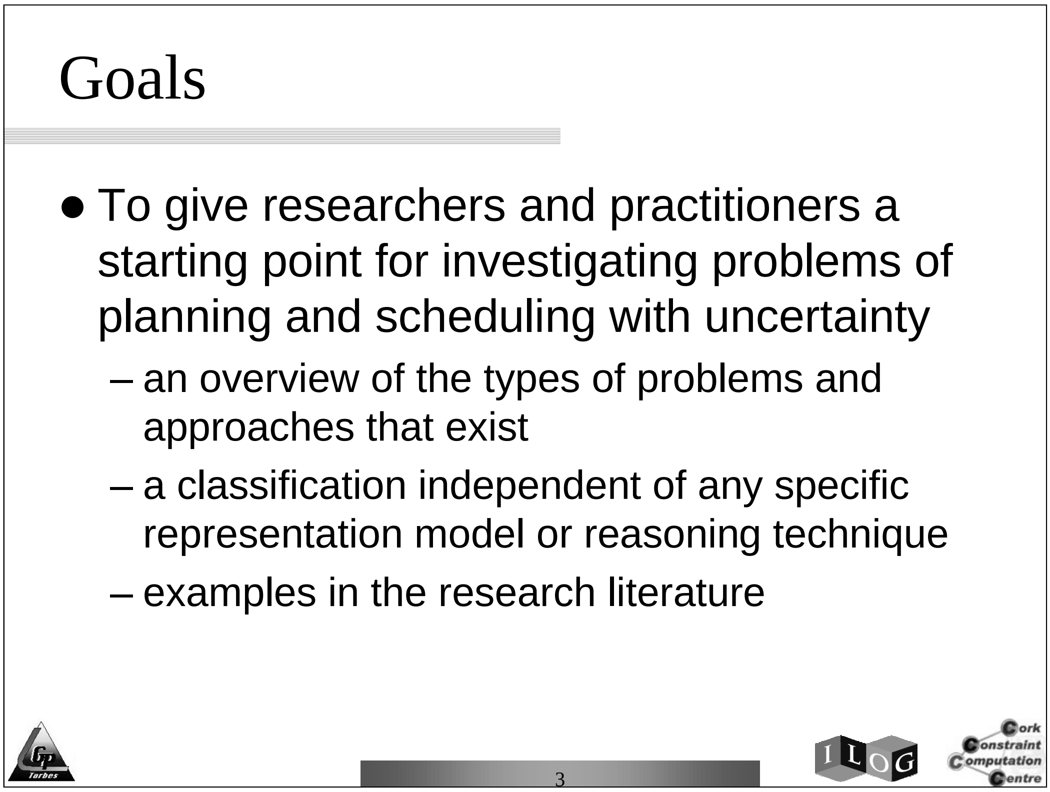
Practical Approaches to Handling Uncertainty in Planning & Scheduling

- **J. Christopher Beck**
  `c.beck@4c.ucc.ie`
- **Thierry Vidal**
  `thierry@enit.fr`



Speakers

- J. Christopher Beck
  - Staff Scientist, Cork Constraint Computation Centre (4C), Ireland
  - Research interests
    - constraint-directed scheduling
    - heuristic search
    - problem structure
    - real world constraints
- Thierry Vidal
  - Associate Professor, LGP / ENIT, Tarbes, France
  - Research interests
    - temporal constraint problems
    - temporal uncertainty
    - conditional planning
    - multi-agent scheduling



Goals

- To give researchers and practitioners a starting point for investigating problems of planning and scheduling with uncertainty
  - an overview of the types of problems and approaches that exist
  - a classification independent of any specific representation model or reasoning technique
  - examples in the research literature

3

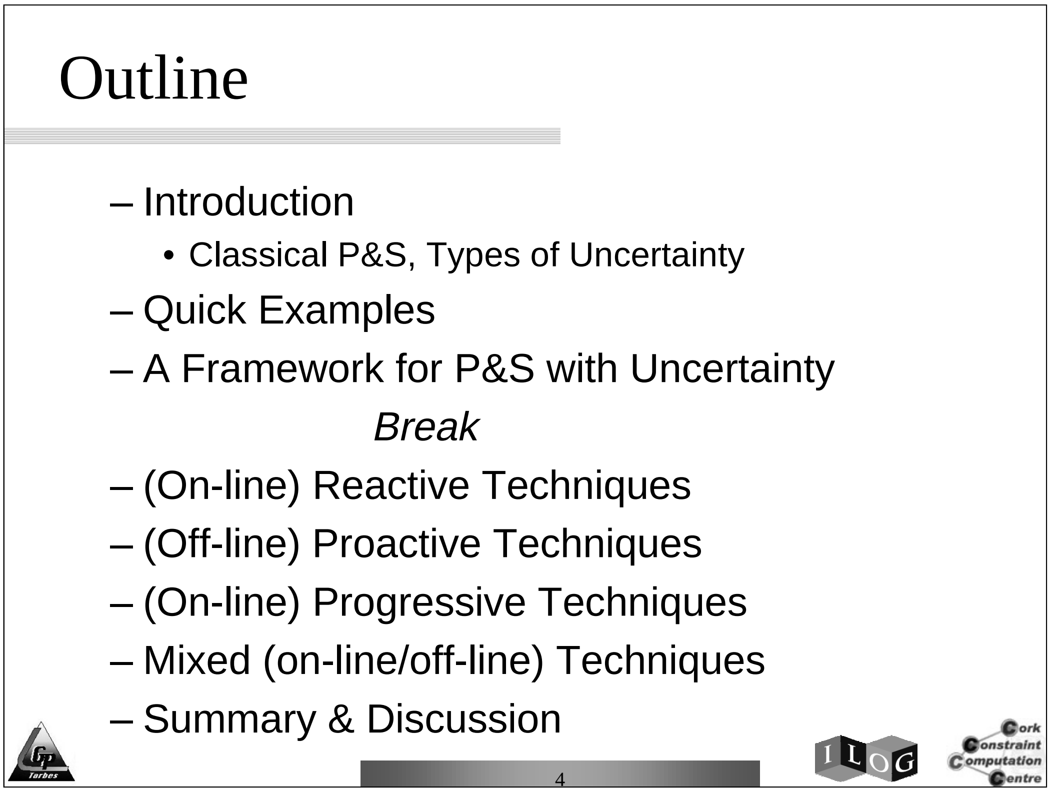
What kinds of problems have been addressed in the literature?

- As we will see there are a wide variety of problem definitions and emphases.

What are the approaches that have been tried and what is their theoretical basis (if any)?

To really solve a full planning/scheduling problem with uncertainty, it is necessary to have an integrated problem-solving-and-execution system with:

- off-line problem solving
- on-line reasoning
- execution monitoring
- communication and coordination among execution and reasoning components



Outline

- Introduction
  - Classical P&S, Types of Uncertainty
- Quick Examples
- A Framework for P&S with Uncertainty

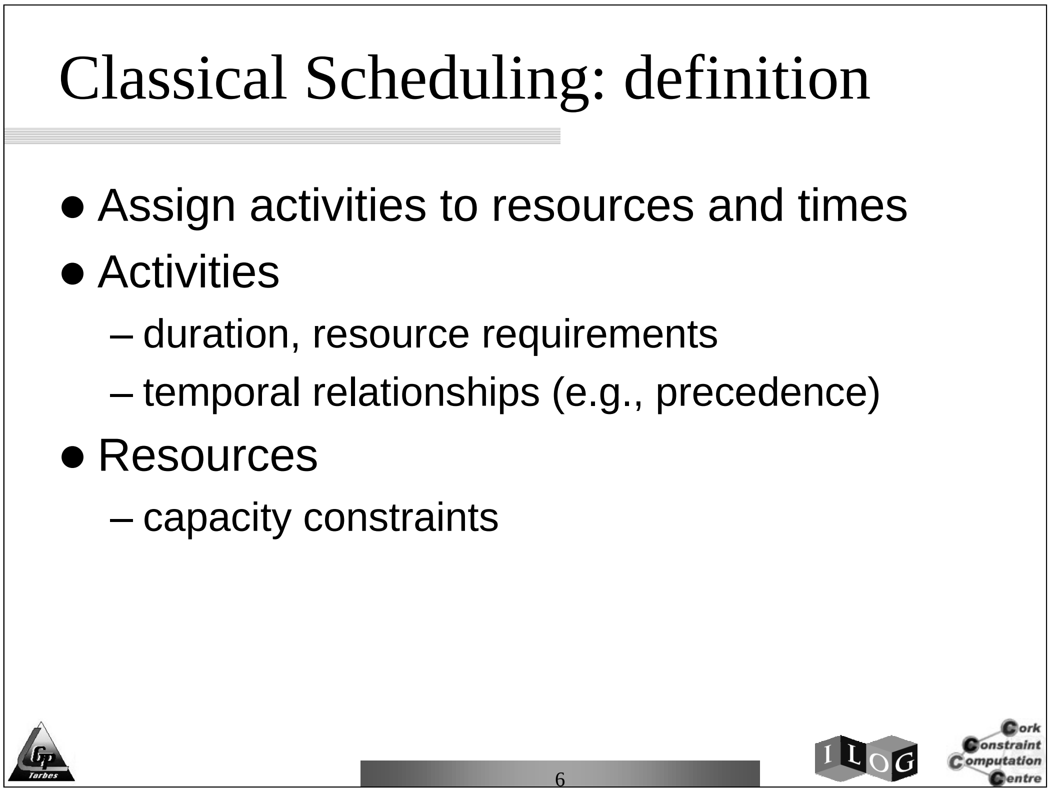
Break

- (On-line) Reactive Techniques
- (Off-line) Proactive Techniques
- (On-line) Progressive Techniques
- Mixed (on-line/off-line) Techniques
- Summary & Discussion

Introduction

Classical Planning & Scheduling

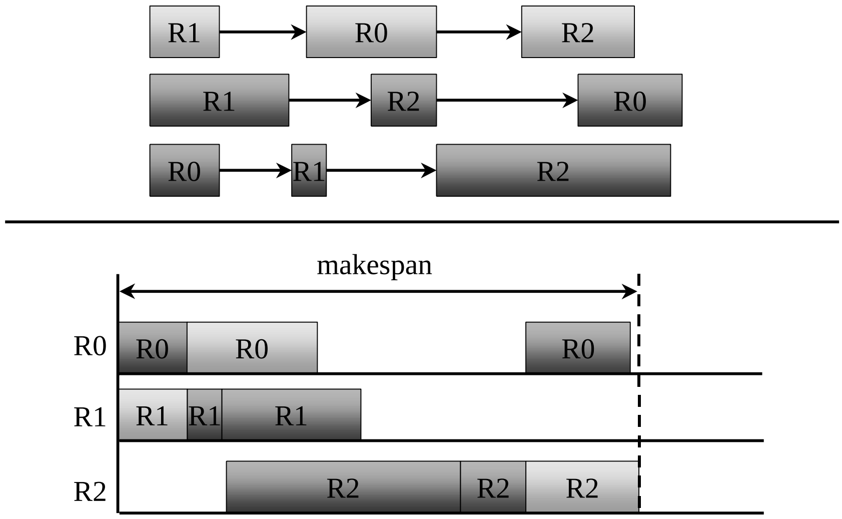
Types of Uncertainty

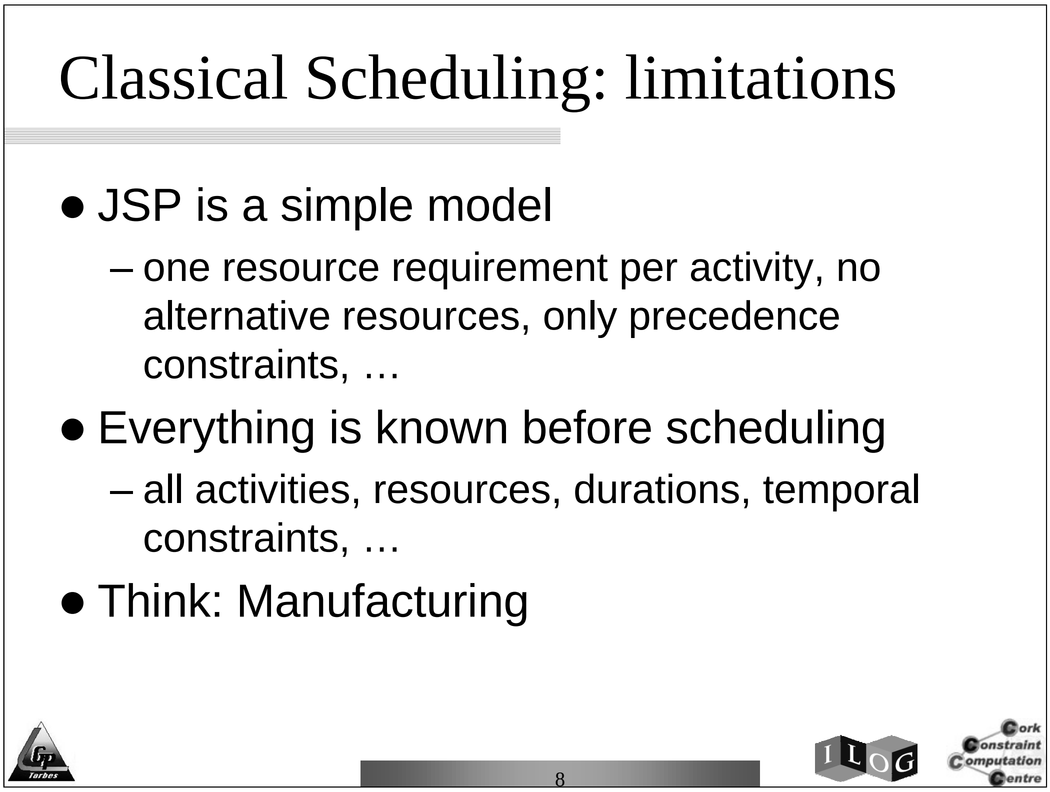


Classical Scheduling: definition

- Assign activities to resources and times
- Activities
  - duration, resource requirements
  - temporal relationships (e.g., precedence)
- Resources
  - capacity constraints

Job Shop Scheduling Problem (JSP)



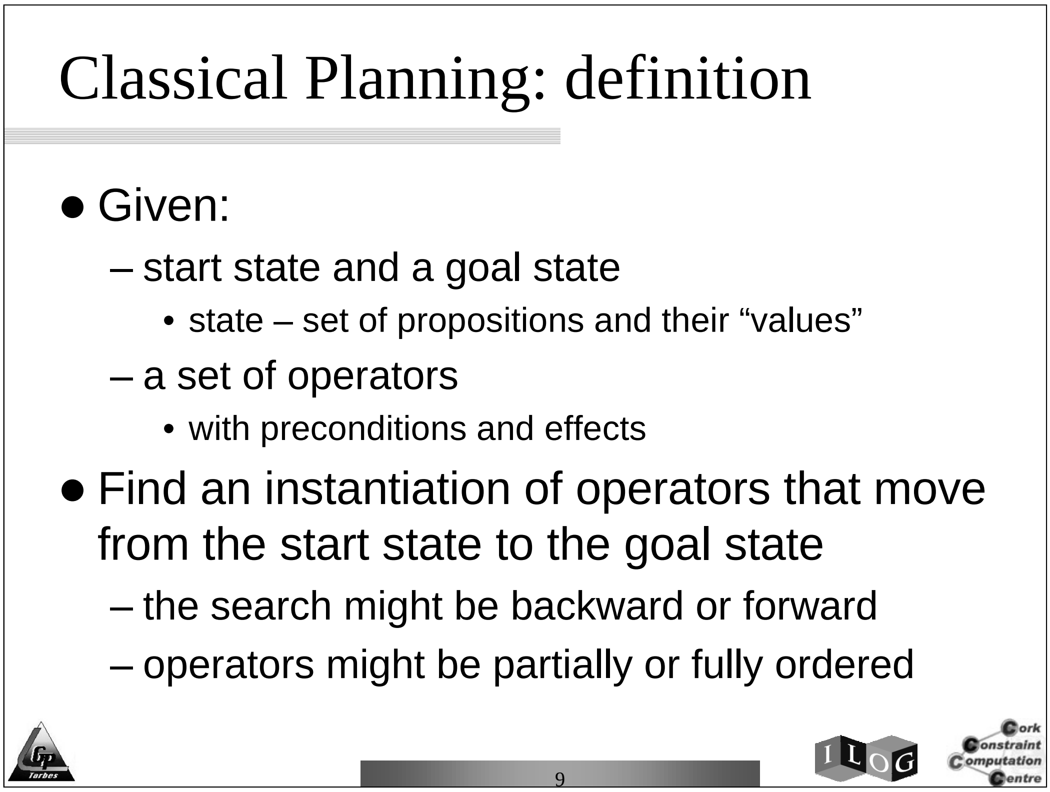


Classical Scheduling: limitations

- JSP is a simple model
  - one resource requirement per activity, no alternative resources, only precedence constraints, …
- Everything is known before scheduling
  - all activities, resources, durations, temporal constraints, …
- Think: Manufacturing

There are, of course, other models of scheduling than JSP (e.g., flow-shop, open-shop, timetabling, RCPSP, …).

The JSP is one of the most common models.



Classical Planning: definition

- Given:
  - start state and a goal state
    - state – set of propositions and their “values”
  - a set of operators
    - with preconditions and effects
- Find an instantiation of operators that move from the start state to the goal state
  - the search might be backward or forward
  - operators might be partially or fully ordered

Forward chaining = the search tree takes the start state as the root and develops branches towards the goal state
Backward chaining = the search tree takes the goal state as the root and develops branches towards the start state

When operators are partially ordered, this is called non-linear planning and enforces more complex requirements on the plan (e.g. the Chapman criteria that demands that if an effect of an action Producer is also the precondition of an action Consumer, then any other action Threat that "consumes" the same precondition must not hold in-between)

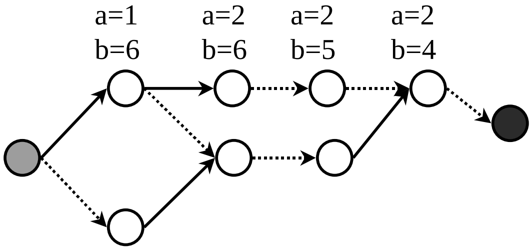
Classical Planning: example

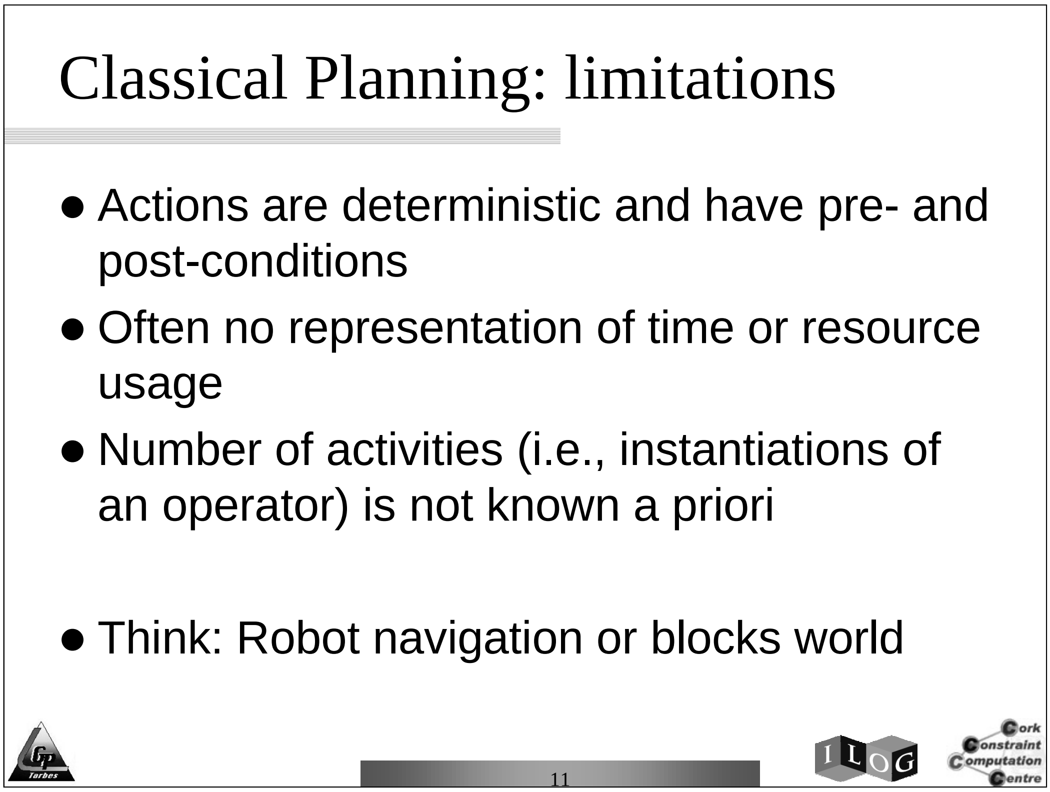
● Start: a = 0, b = 6

● Goal: |b – a| ≤ 1

Operator 1: (a < 3) ⟶ (a = a+1)

Operator 2: (b > 3) ⇢ (b = b-1)





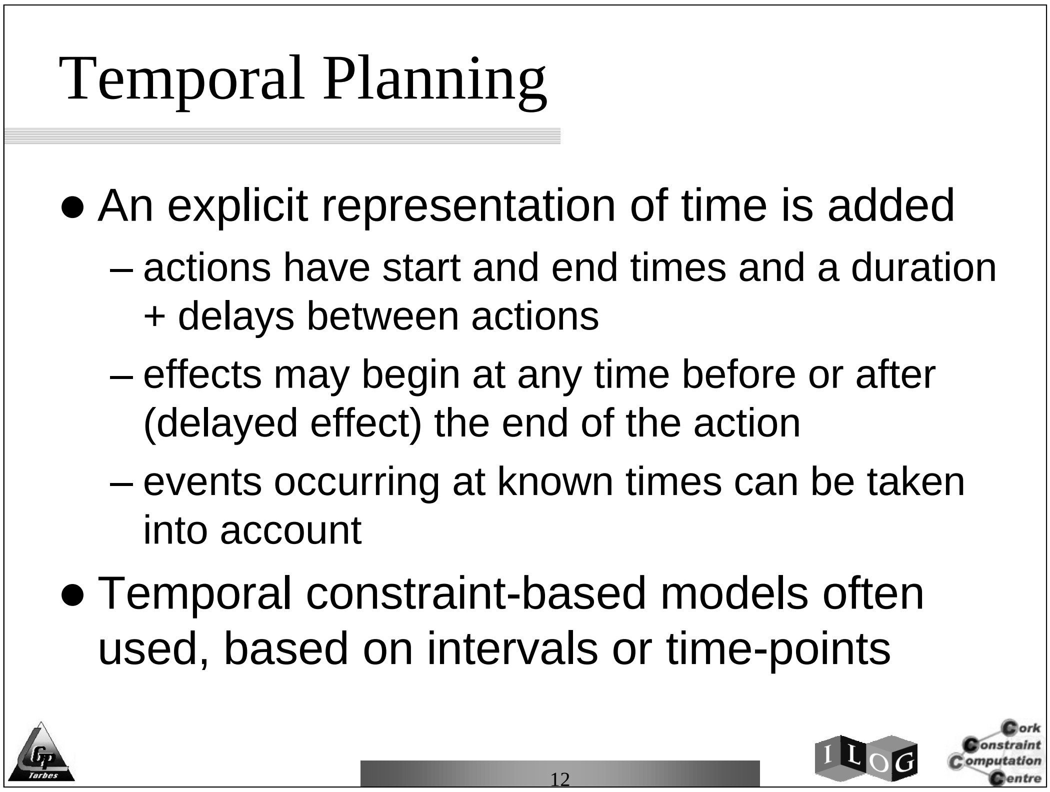
Classical Planning: limitations

- Actions are deterministic and have pre- and post-conditions
- Often no representation of time or resource usage
- Number of activities (i.e., instantiations of an operator) is not known a priori

- Think: Robot navigation or blocks world

The most well-known of classical planners is Graphplan.

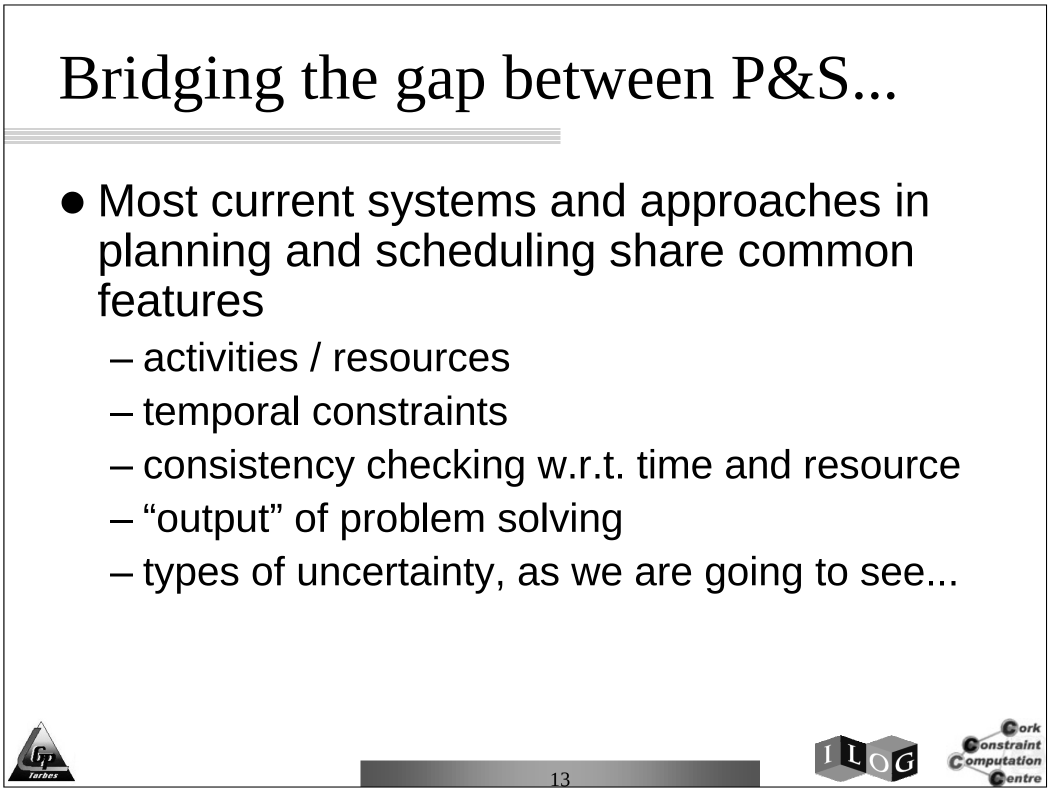
We will focus on *task-oriented* planning (a task must be performed to reach a fixed goal) as opposed to *process-oriented* planning (a process must be controlled so that fixed conditions are maintained).



Temporal Planning

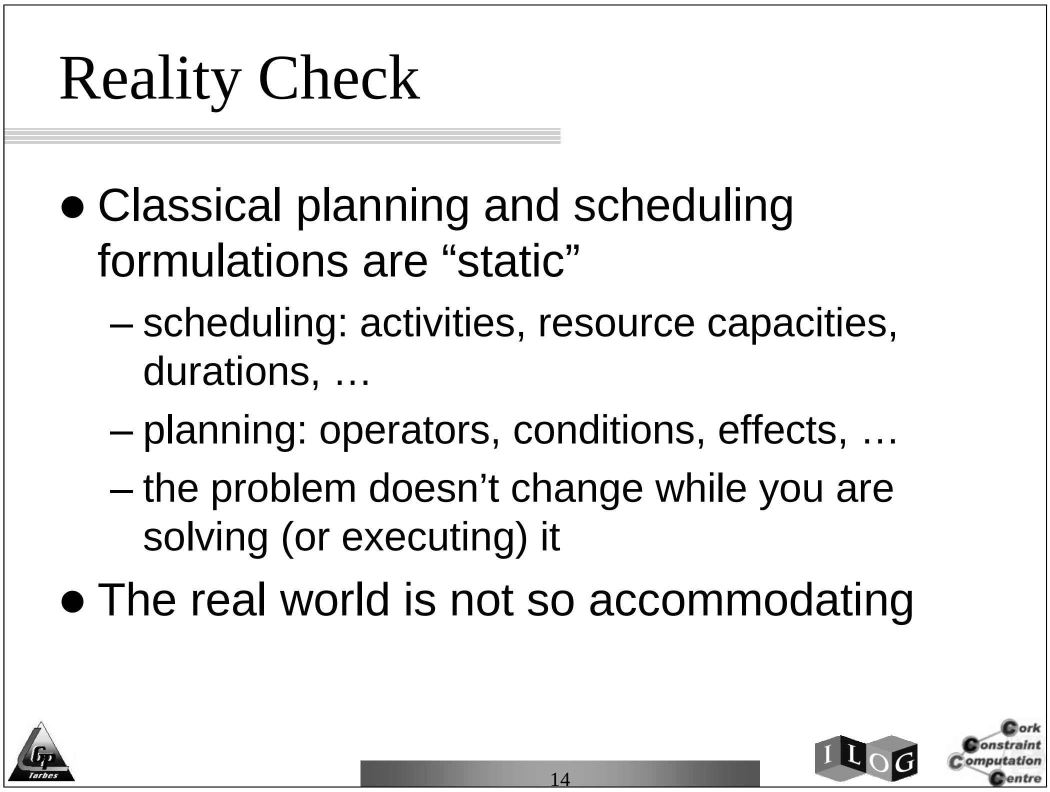
- An explicit representation of time is added
  - – actions have start and end times and a duration + delays between actions
  - – effects may begin at any time before or after (delayed effect) the end of the action
  - – events occurring at known times can be taken into account
- Temporal constraint-based models often used, based on intervals or time-points

IxTeT (LAAS-CNRS) and HSTS (planner in NASA project Remote Experiment) are the main examples of such temporal planners.



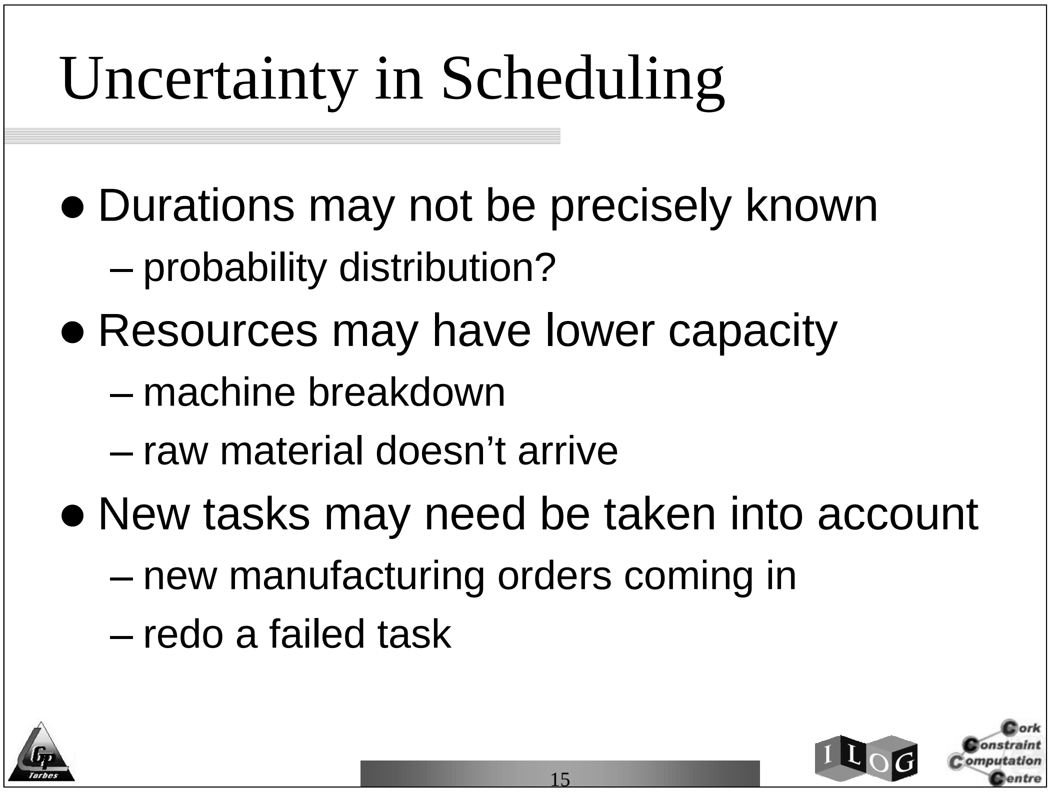
Bridging the gap between P&S...

- Most current systems and approaches in planning and scheduling share common features
  - activities / resources
  - temporal constraints
  - consistency checking w.r.t. time and resource
  - “output” of problem solving
  - types of uncertainty, as we are going to see...



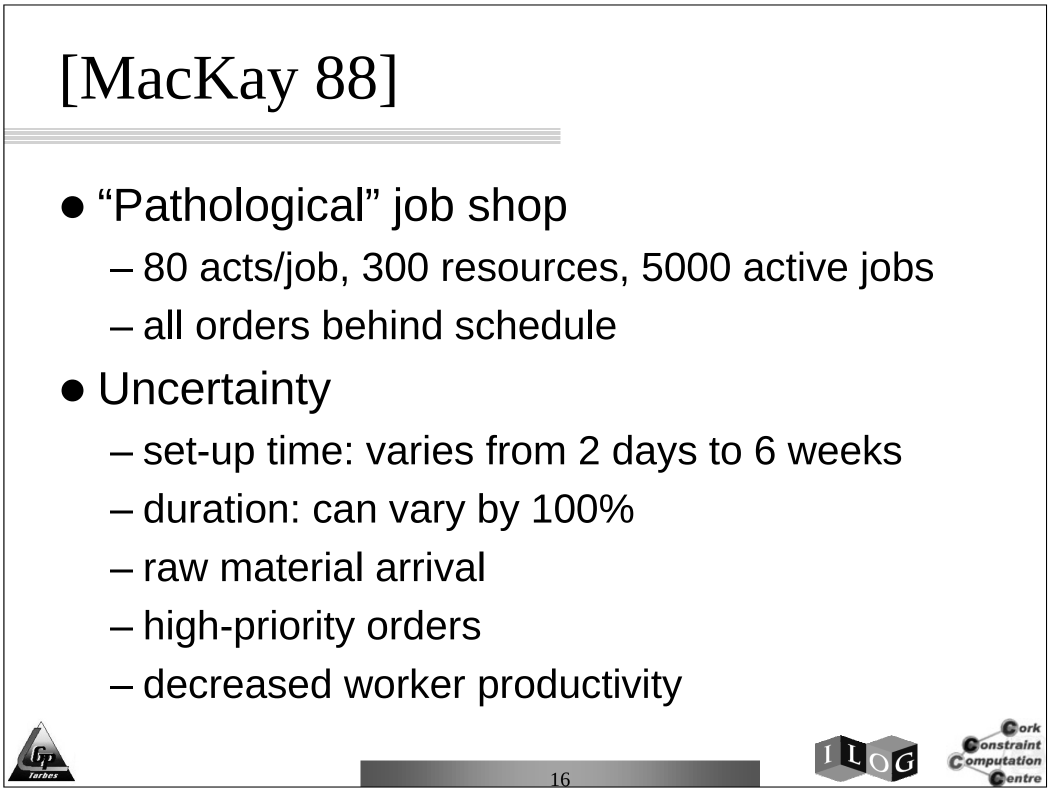
Reality Check

- Classical planning and scheduling formulations are “static”
  - scheduling: activities, resource capacities, durations, …
  - planning: operators, conditions, effects, …
  - the problem doesn’t change while you are solving (or executing) it
- The real world is not so accommodating



Uncertainty in Scheduling

- Durations may not be precisely known
  - – probability distribution?
- Resources may have lower capacity
  - – machine breakdown
  - – raw material doesn’t arrive
- New tasks may need be taken into account
  - – new manufacturing orders coming in
  - – redo a failed task



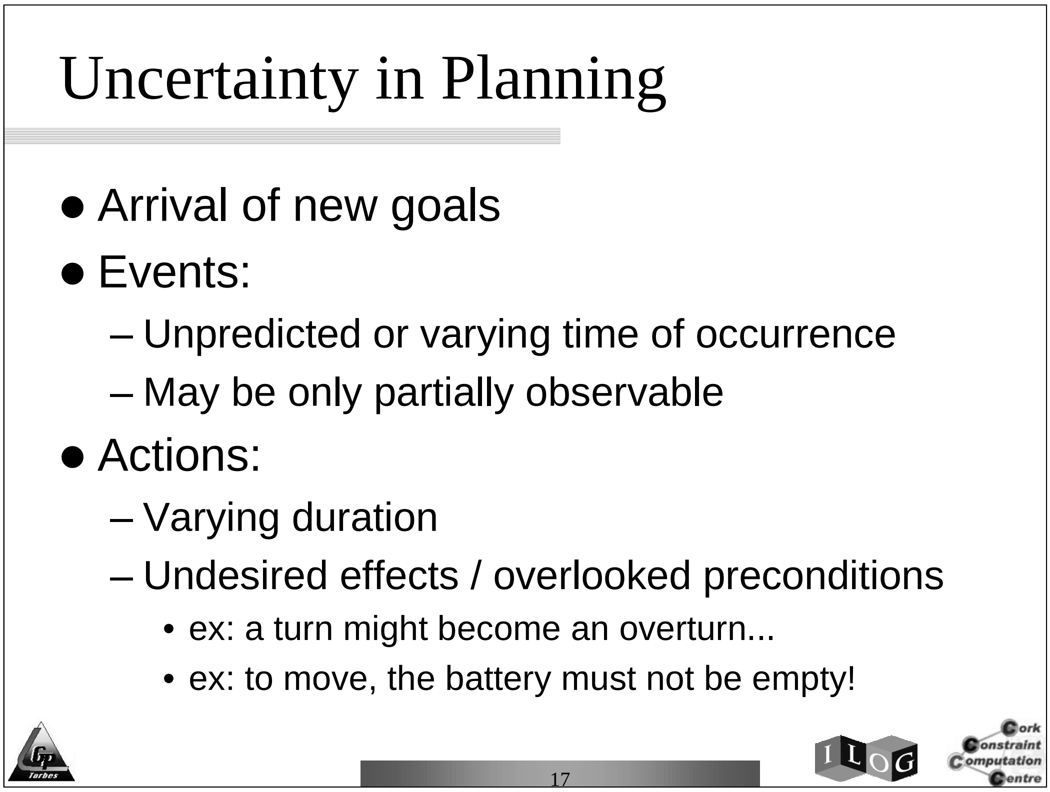
[MacKay 88]

- "Pathological" job shop
  - – 80 acts/job, 300 resources, 5000 active jobs
  - – all orders behind schedule
- Uncertainty
  - – set-up time: varies from 2 days to 6 weeks
  - – duration: can vary by 100%
  - – raw material arrival
  - – high-priority orders
  - – decreased worker productivity

16

[MacKay 88] MacKay, K.N., Safayeni, F.R. & Buzacott, J.A. Job-shop scheduling theory: What is relevant? *Interfaces*, 18(4): 84-90, 1988.

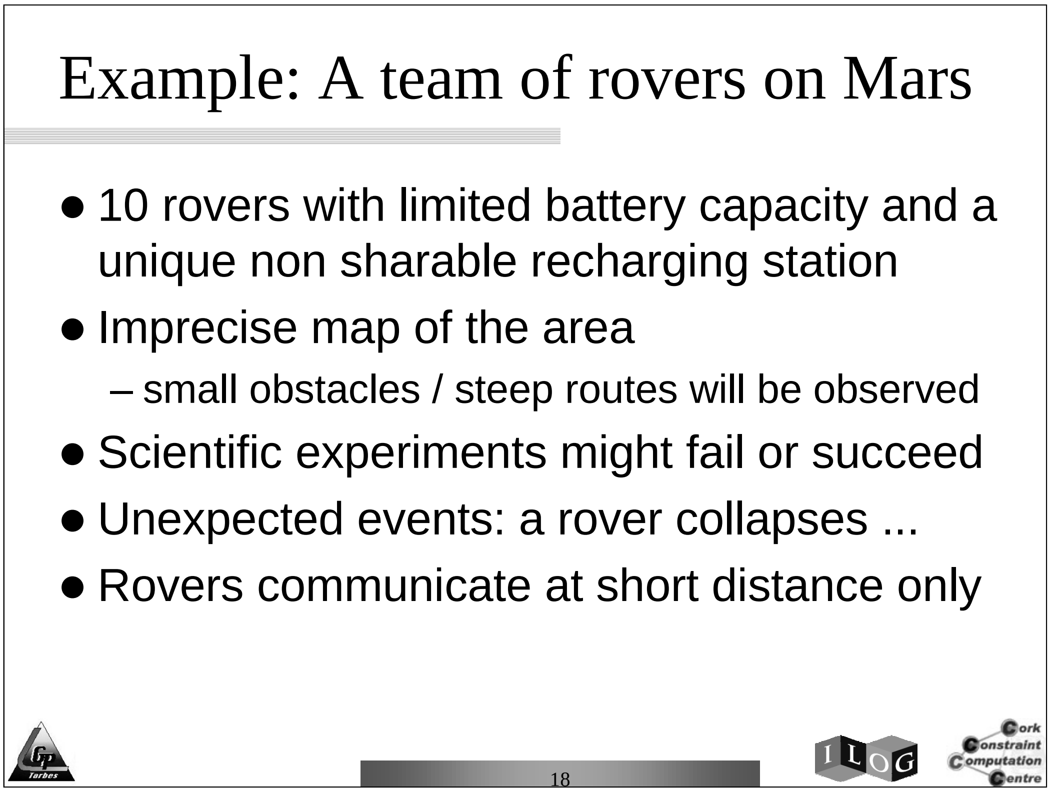
Set-up time: some time is necessary to configure a resource to be able to process an activity. The length of the setup may depend on the preceding and following activities. The classical example is paint mixing: if you switch from mixing black paint to white paint, you need to completely clean the machine. Going from white to grey requires less of a cleaning effort and therefore less time for a setup. With sophisticated parameterizable machinery, scheduling to minimize setup time is a common optimization criterion.



Uncertainty in Planning

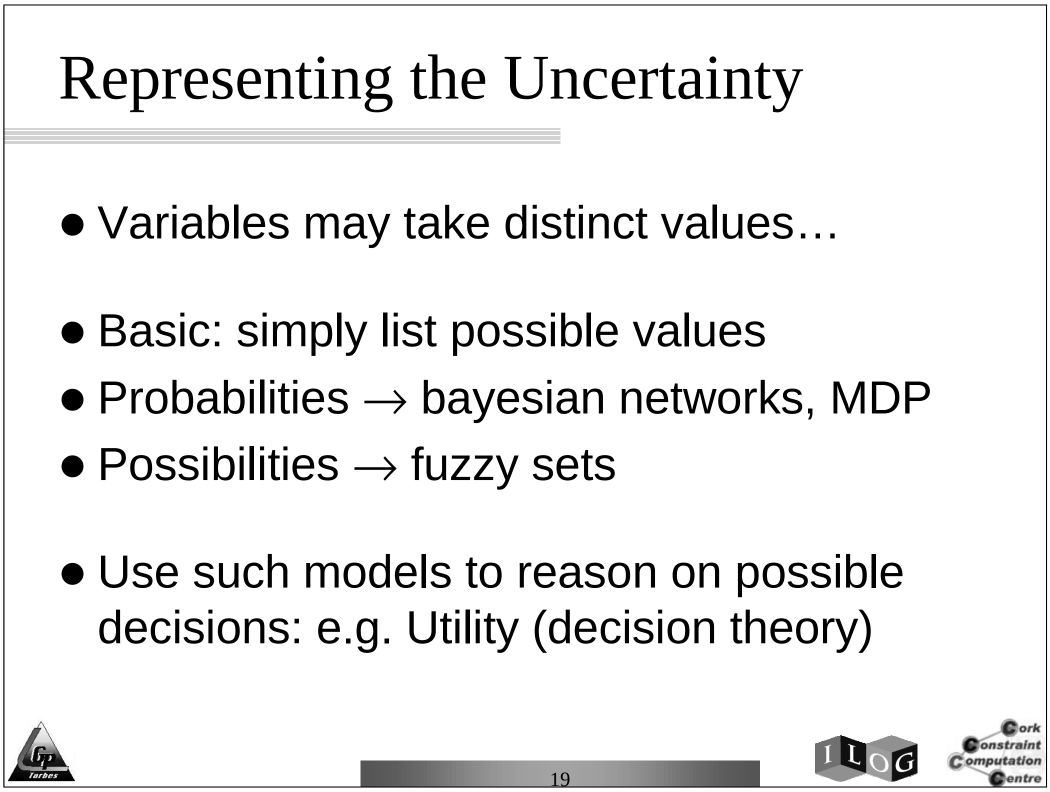
- Arrival of new goals
- Events:
  - – Unpredicted or varying time of occurrence
  - – May be only partially observable
- Actions:
  - – Varying duration
  - – Undesired effects / overlooked preconditions
    - ex: a turn might become an overturn...
    - ex: to move, the battery must not be empty!

In this tutorial we will not address uncertainty in the sensing. That means the executing agent has *full observability* on the environment and always knows exactly which state it is in.



Example: A team of rovers on Mars

- 10 rovers with limited battery capacity and a unique non sharable recharging station
- Imprecise map of the area
  - small obstacles / steep routes will be observed
- Scientific experiments might fail or succeed
- Unexpected events: a rover collapses ...
- Rovers communicate at short distance only



Representing the Uncertainty

- Variables may take distinct values…
- Basic: simply list possible values
- Probabilities → bayesian networks, MDP
- Possibilities → fuzzy sets
- Use such models to reason on possible decisions: e.g. Utility (decision theory)

- Basic "disjunctive" models = just distinguish between possible cases (discrete or continuous) that may arise at execution time:

  - intervals of possible durations: $d\in [l, u]$
  - different states a resource might be in: $Ok \vee Fail$
  - distinct outcomes an action might have: $E_1 \vee E_2$

No ranking : same probability assumed for all possible cases (or possibility of 1 for each case).

- Probabilities or possibilities?

  Probabilities = for each possible case, statistical data (i.e. precise and reliable numbers) are available

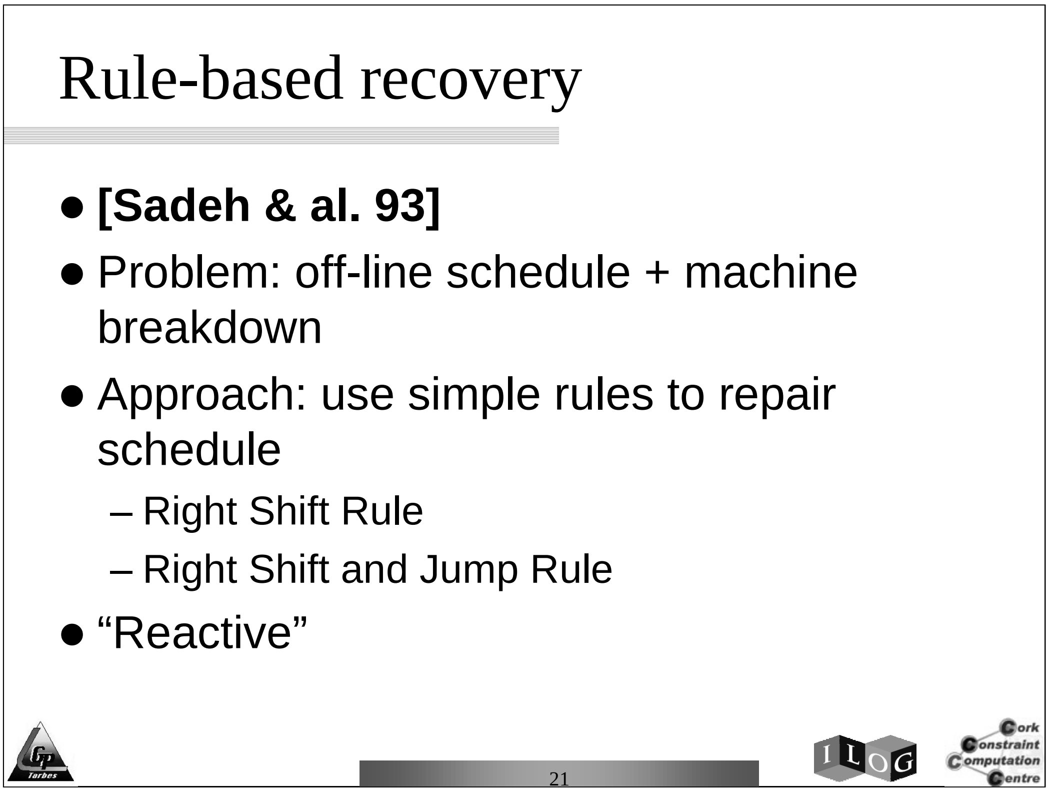
  Fuzzy sets = qualitative or subjective + partial knowledge: even full ignorance might be expressed!

  ex: we don't know if a patient is sick or not

  with probabilities: $P(ok)=P(sick)=1/2$ *?*
  $P(ok)=P(benign)=P(acute)=1/3$ *?…*

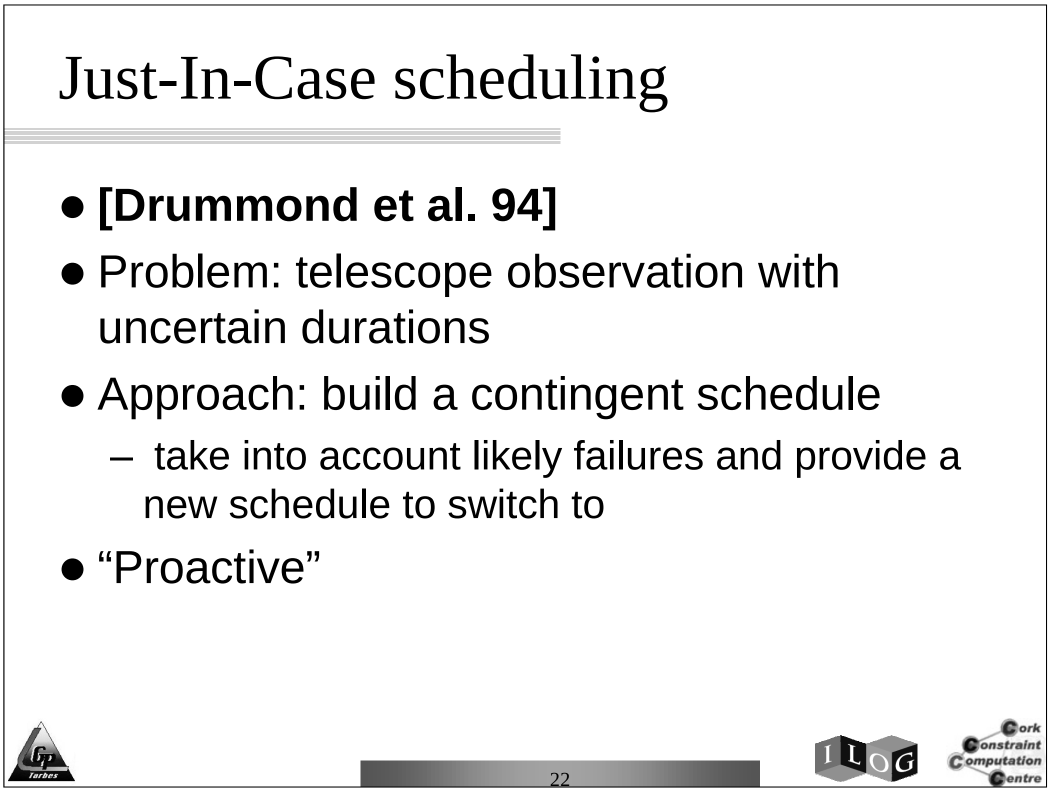
  with possibilities: $\Pi(ok)= \Pi(sick)=1$ *and* $N(ok)=N(sick)=0$

Quick Examples



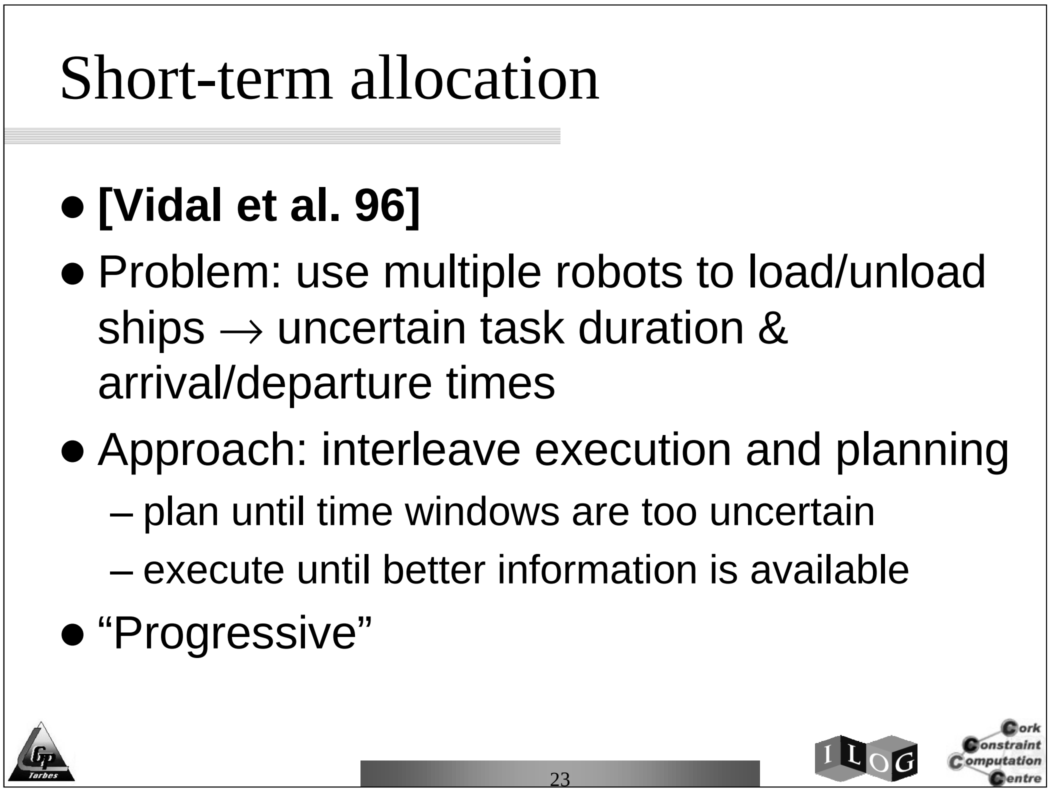
Rule-based recovery

- **[Sadeh & al. 93]**
- Problem: off-line schedule + machine breakdown
- Approach: use simple rules to repair schedule
  - Right Shift Rule
  - Right Shift and Jump Rule
- “Reactive”



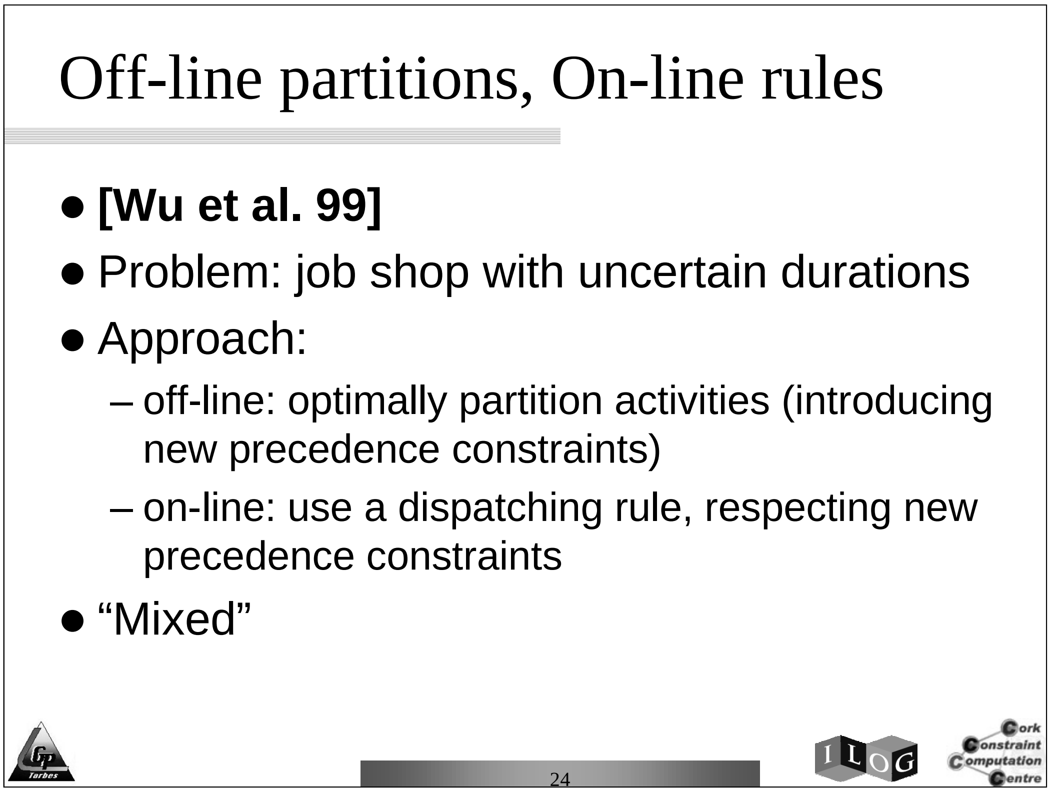
Just-In-Case scheduling

- **[Drummond et al. 94]**
- Problem: telescope observation with uncertain durations
- Approach: build a contingent schedule
  - take into account likely failures and provide a new schedule to switch to
- “Proactive”



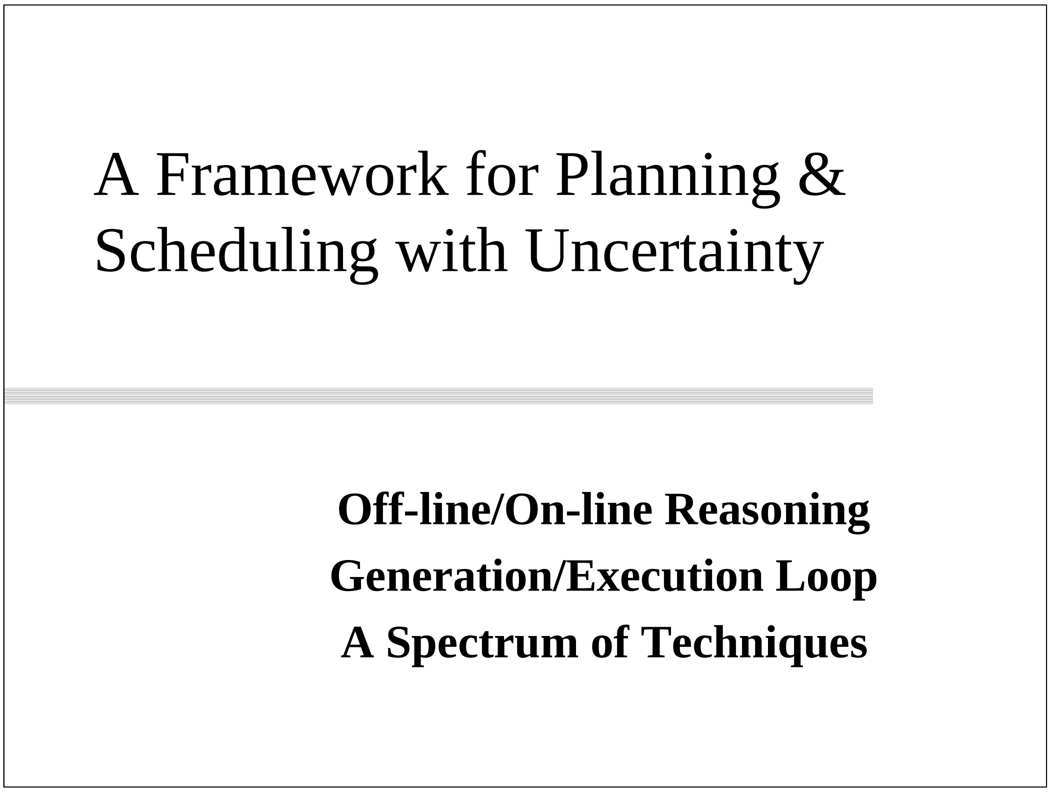
Short-term allocation

- **[Vidal et al. 96]**
- Problem: use multiple robots to load/unload ships → uncertain task duration & arrival/departure times
- Approach: interleave execution and planning
  - plan until time windows are too uncertain
  - execute until better information is available
- “Progressive”



Off-line partitions, On-line rules

- **[Wu et al. 99]**
- Problem: job shop with uncertain durations
- Approach:
  - off-line: optimally partition activities (introducing new precedence constraints)
  - on-line: use a dispatching rule, respecting new precedence constraints
- “Mixed”

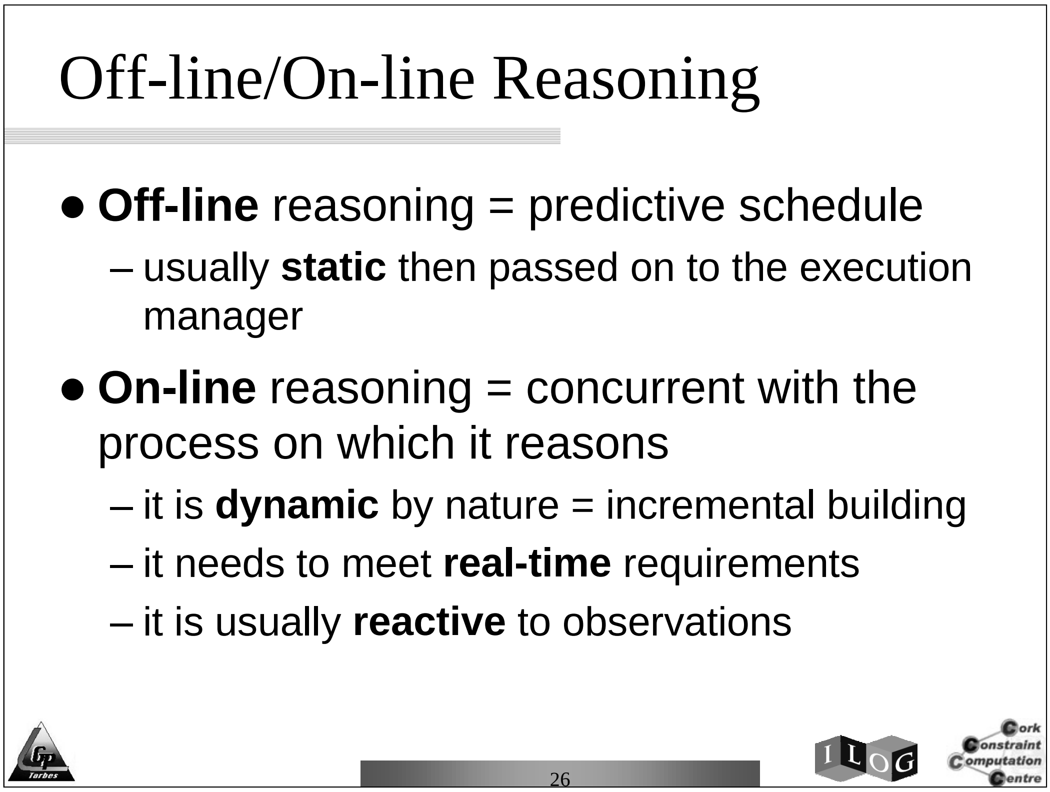


A Framework for Planning & Scheduling with Uncertainty

Off-line/On-line Reasoning

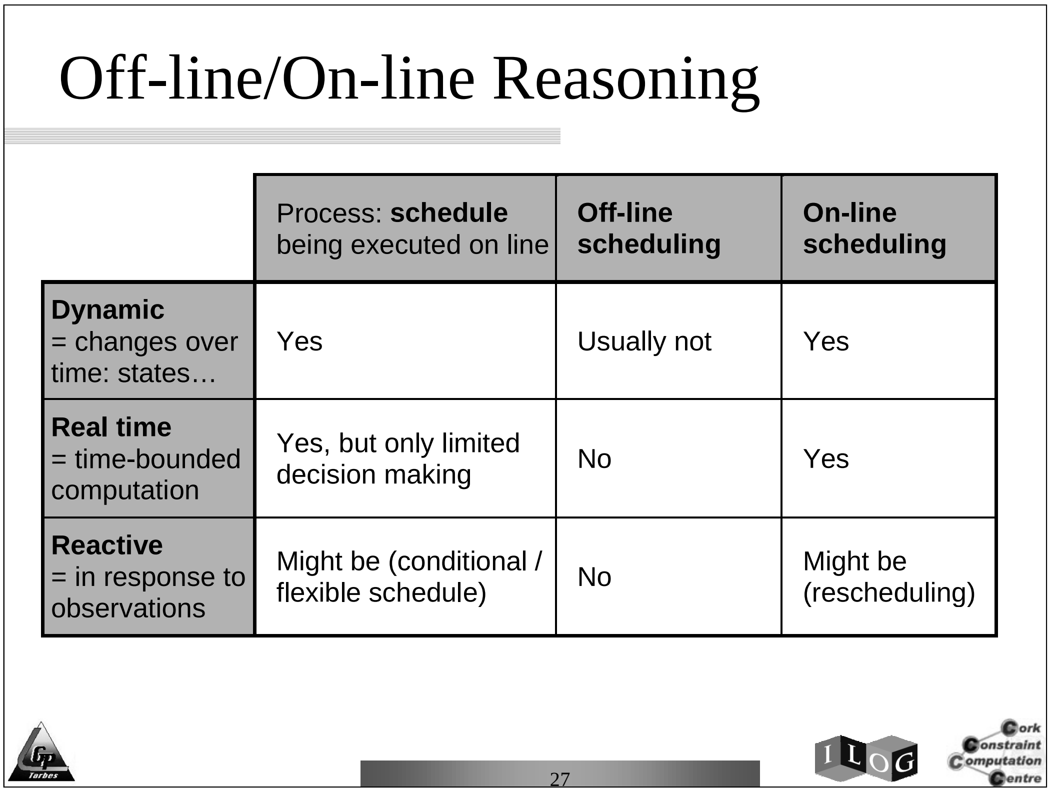
Generation/Execution Loop

A Spectrum of Techniques



Off-line/On-line Reasoning

- **Off-line** reasoning = predictive schedule
  - usually **static** then passed on to the execution manager
- **On-line** reasoning = concurrent with the process on which it reasons
  - it is **dynamic** by nature = incremental building
  - it needs to meet **real-time** requirements
  - it is usually **reactive** to observations

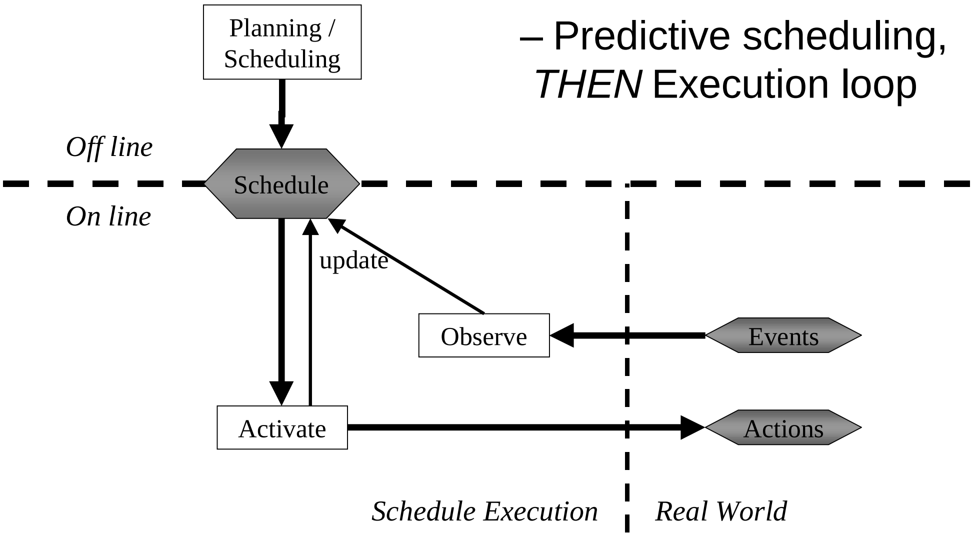


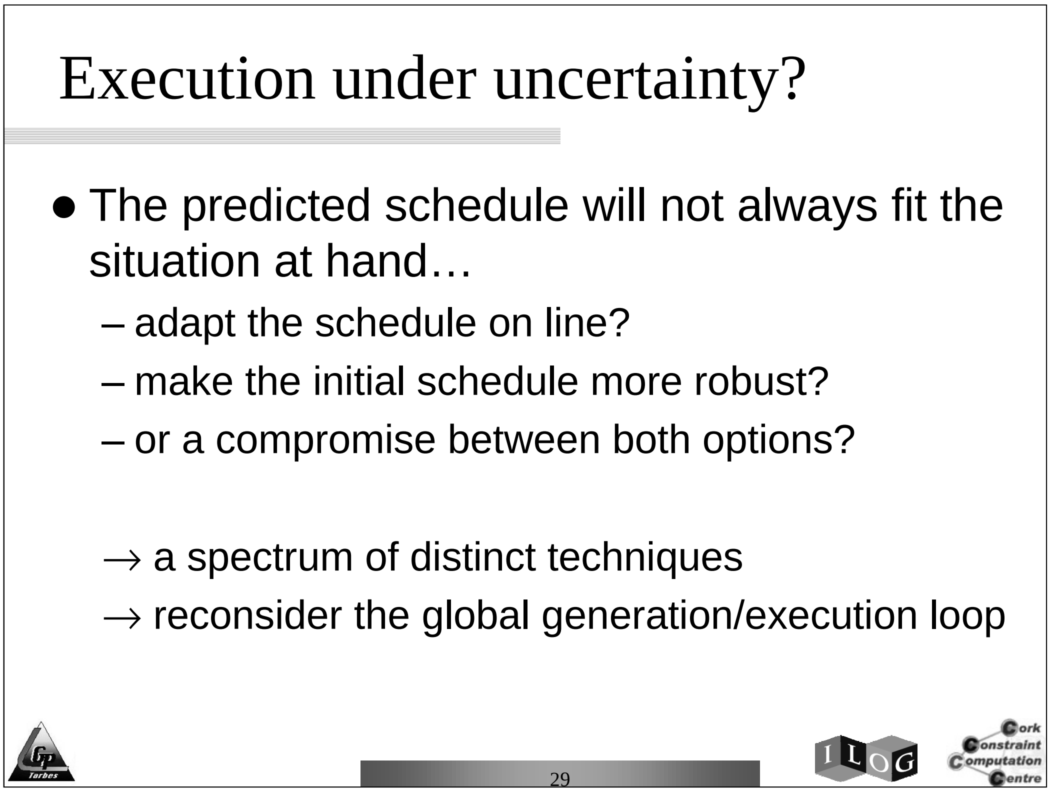
Off-line/On-line Reasoning

	Process: **schedule** being executed on line	**Off-line scheduling**	**On-line scheduling**
Dynamic = changes over time: states…	Yes	Usually not	Yes
Real time = time-bounded computation	Yes, but only limited decision making	No	Yes
Reactive = in response to observations	Might be (conditional / flexible schedule)	No	Might be (rescheduling)

Execution in the ideal world

– Predictive scheduling, *THEN* Execution loop



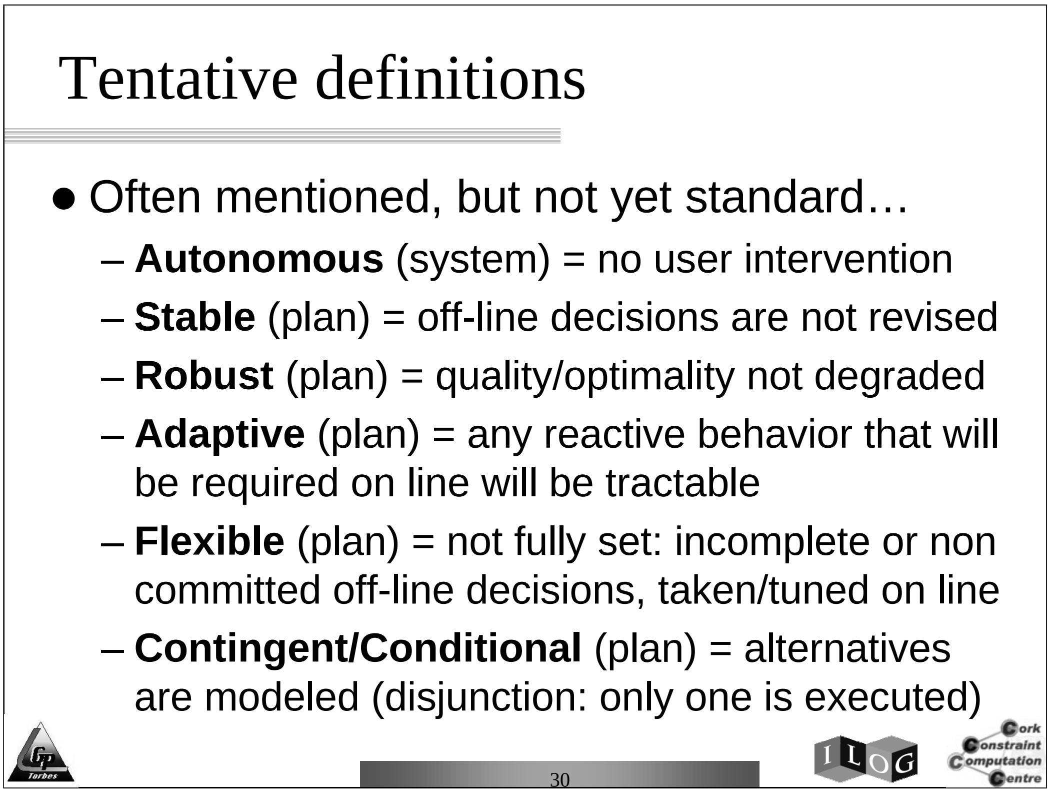


Execution under uncertainty?

- The predicted schedule will not always fit the situation at hand…
  - – adapt the schedule on line?
  - – make the initial schedule more robust?
  - – or a compromise between both options?

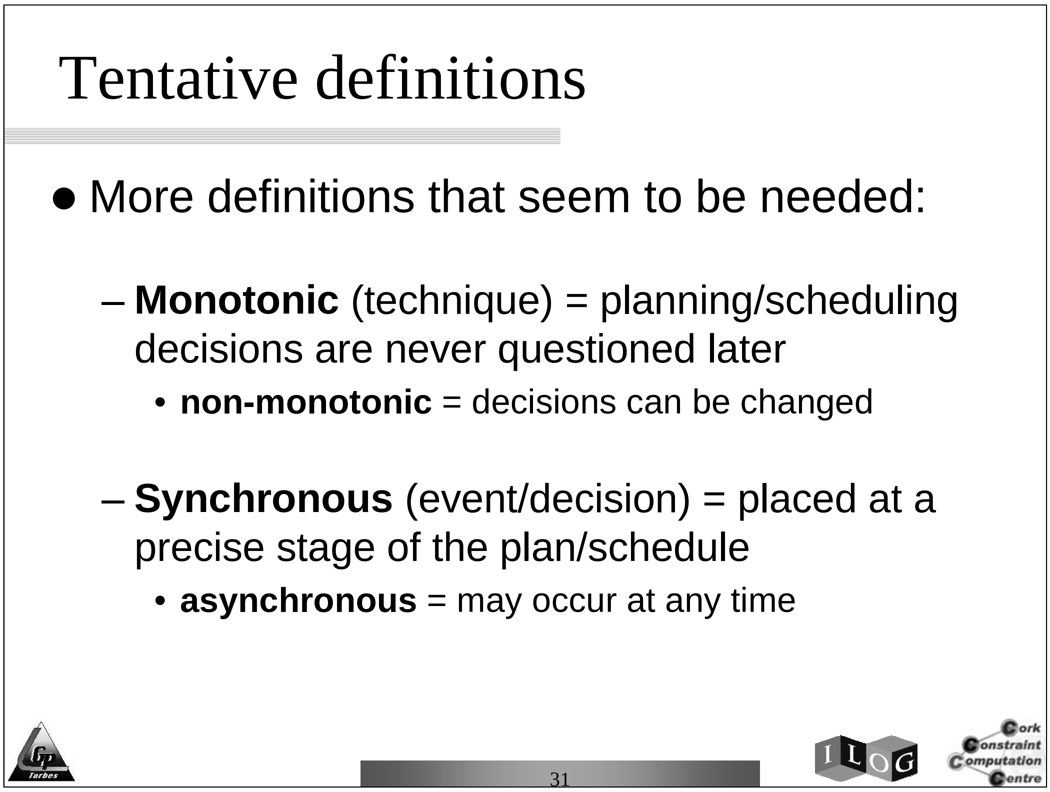
→ a spectrum of distinct techniques

→ reconsider the global generation/execution loop



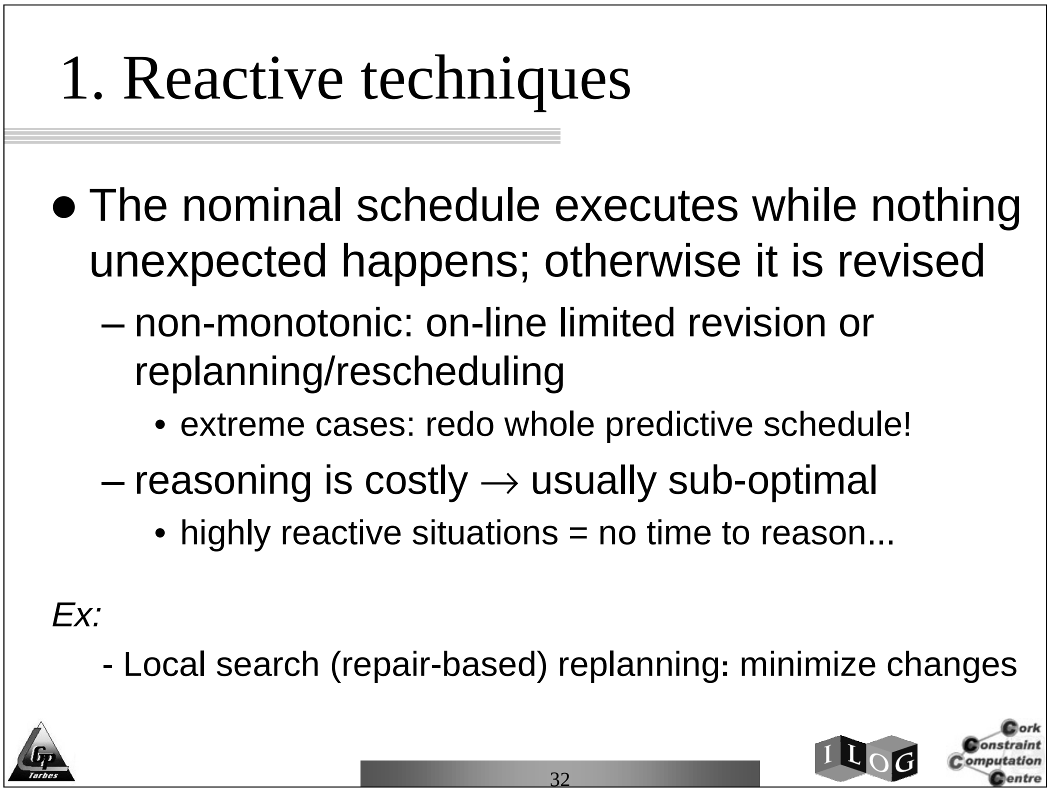
Tentative definitions

- Often mentioned, but not yet standard…
  - **Autonomous** (system) = no user intervention
  - **Stable** (plan) = off-line decisions are not revised
  - **Robust** (plan) = quality/optimality not degraded
  - **Adaptive** (plan) = any reactive behavior that will be required on line will be tractable
  - **Flexible** (plan) = not fully set: incomplete or non committed off-line decisions, taken/tuned on line
  - **Contingent/Conditional** (plan) = alternatives are modeled (disjunction: only one is executed)



Tentative definitions

- More definitions that seem to be needed:
  - **Monotonic** (technique) = planning/scheduling decisions are never questioned later
    - **non-monotonic** = decisions can be changed
  - **Synchronous** (event/decision) = placed at a precise stage of the plan/schedule
    - **asynchronous** = may occur at any time



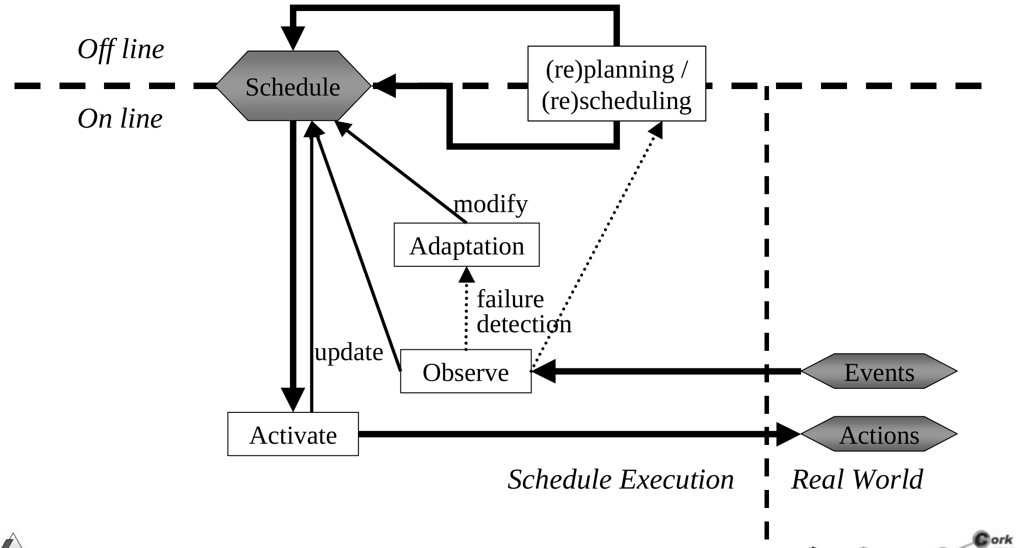
1. Reactive techniques

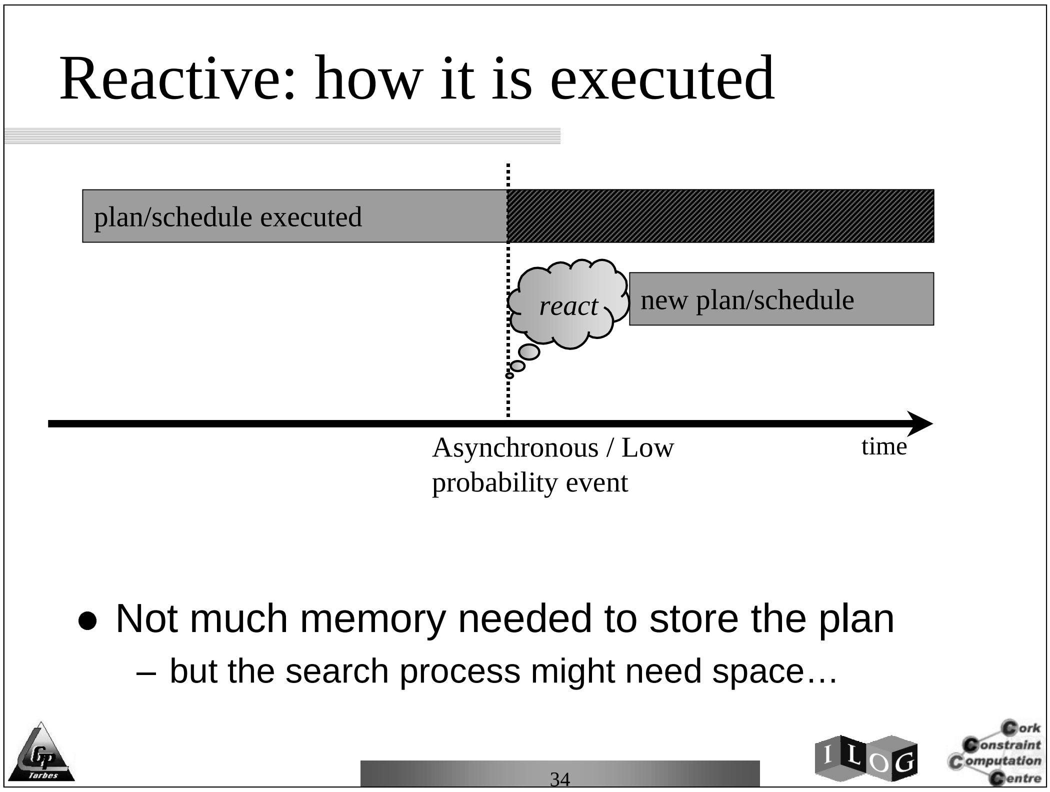
- The nominal schedule executes while nothing unexpected happens; otherwise it is revised
  - non-monotonic: on-line limited revision or replanning/rescheduling
    - extreme cases: redo whole predictive schedule!
  - reasoning is costly → usually sub-optimal
    - highly reactive situations = no time to reason...

Ex:

- Local search (repair-based) replanning**:** minimize changes

Reactive: system architecture





Reactive: how it is executed

plan/schedule executed

react

new plan/schedule

Asynchronous / Low probability event

time

- Not much memory needed to store the plan
  - but the search process might need space…

To summarize:

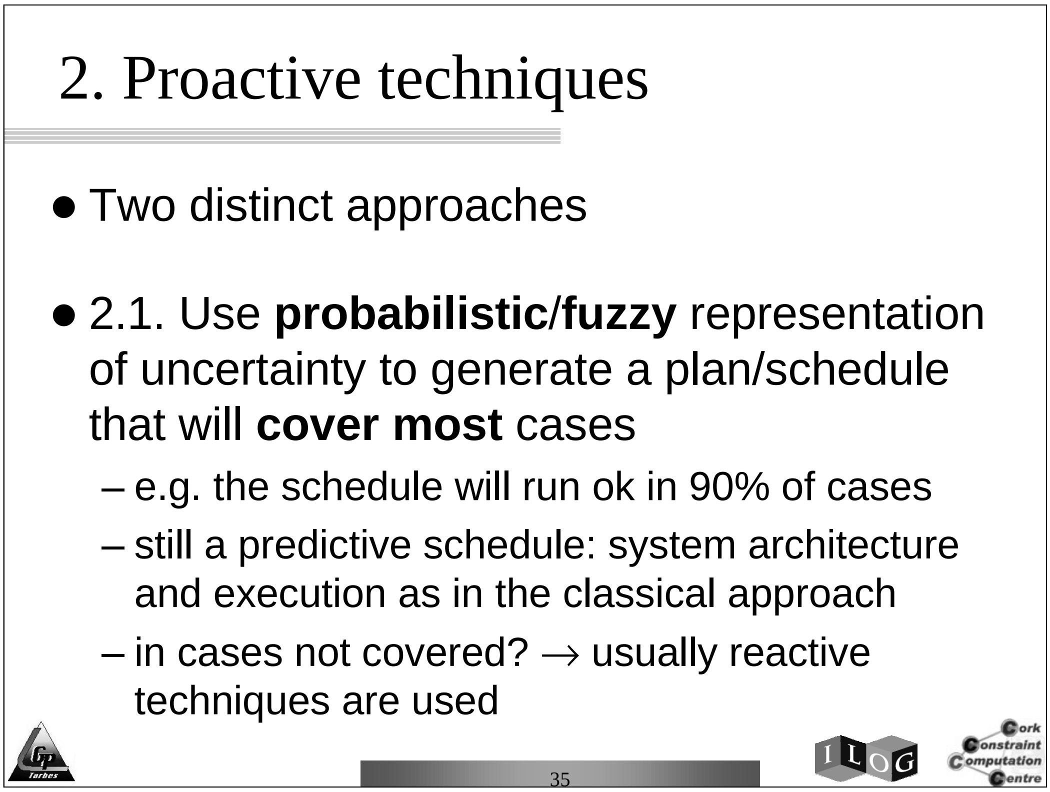
- planning/scheduling decisions must be **quick**;

- hence resulting effective plans/schedules are **sub-optimal**;

- it is a **non-monotonic** technique;

- it is relevant for **asynchronous** and/or **low probability** perturbations;

- it only requires **limited memory** to store the plan, but may require **extended memory** for on-line search.



2. Proactive techniques

- Two distinct approaches
- 2.1. Use **probabilistic/fuzzy** representation of uncertainty to generate a plan/schedule that will **cover most** cases
  - e.g. the schedule will run ok in 90% of cases
  - still a predictive schedule: system architecture and execution as in the classical approach
  - in cases not covered? → usually reactive techniques are used

To summarize:

For all proactive techniques:

- most decisions are taken off line: **no need to be quick**.

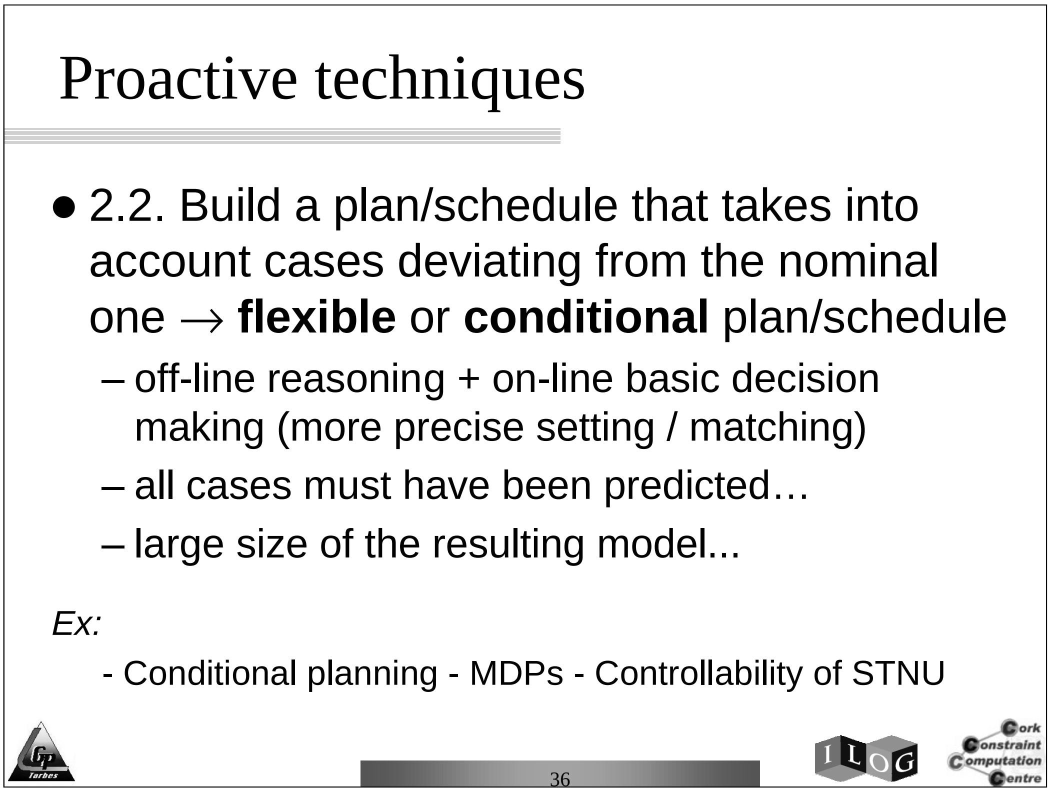
For 'maximum coverage' techniques: they combine predictive and reactive, therefore:

- executed plans/schedules are **sub-optimal**: a compromise is chosen for covering more cases, and for non-covered cases the need for rescheduling can usually be anticipated therefore one still has reasonable time to search;

- it is **monotonic** except for non-covered cases;

- only non-covered cases may occur as **asynchronous** perturbations;

- only **limited memory** required, except for the search in non-covered cases.



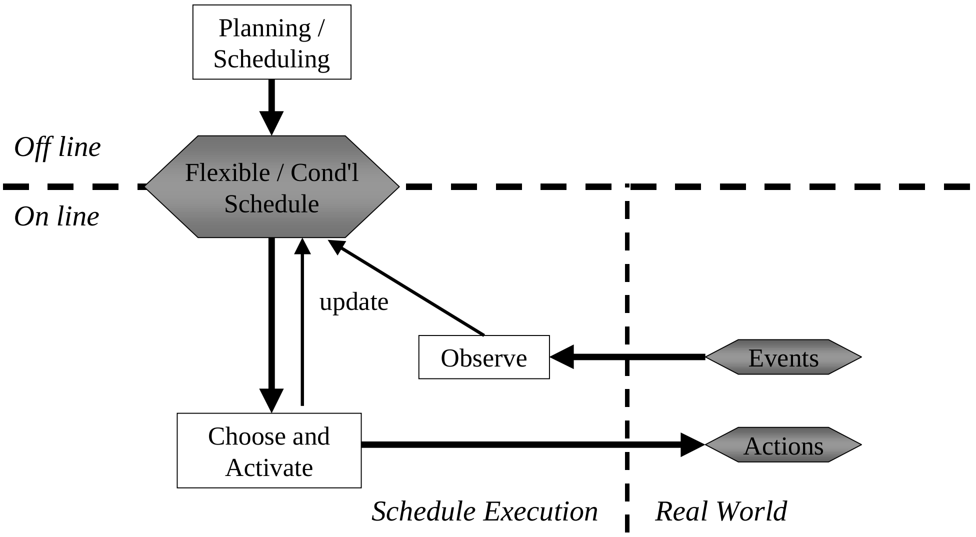
Proactive techniques

- 2.2. Build a plan/schedule that takes into account cases deviating from the nominal one → **flexible** or **conditional** plan/schedule
  - off-line reasoning + on-line basic decision making (more precise setting / matching)
  - all cases must have been predicted…
  - large size of the resulting model...

Ex:

- Conditional planning - MDPs - Controllability of STNU

Proactive: system architecture


Planning / Scheduling

Off line

Flexible / Cond'l Schedule

On line

update

Observe

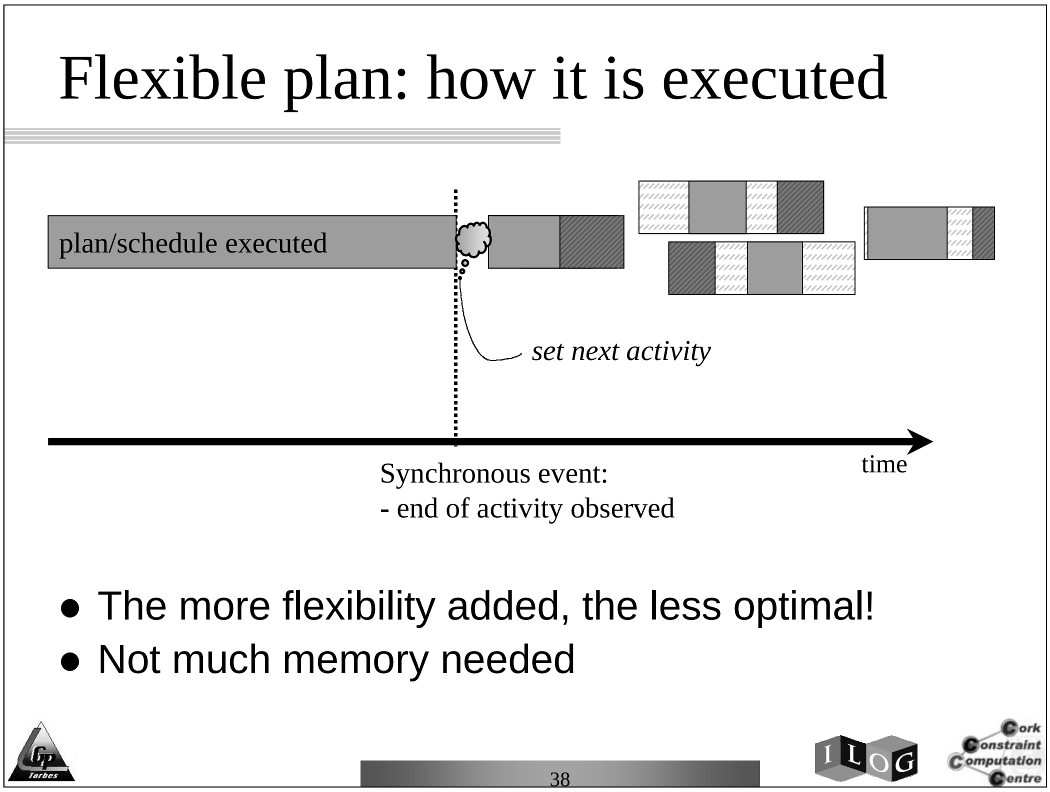
Events

Choose and Activate

Actions

Schedule Execution

Real World



Flexible plan: how it is executed

- The more flexibility added, the less optimal!
- Not much memory needed

To summarize:

- may be **sub-optimal** because it is also somehow a compromise;

- it is a **monotonic** technique;

- it copes with **synchronous** events: each time an observation brings some new information allowing to take a pending decision;

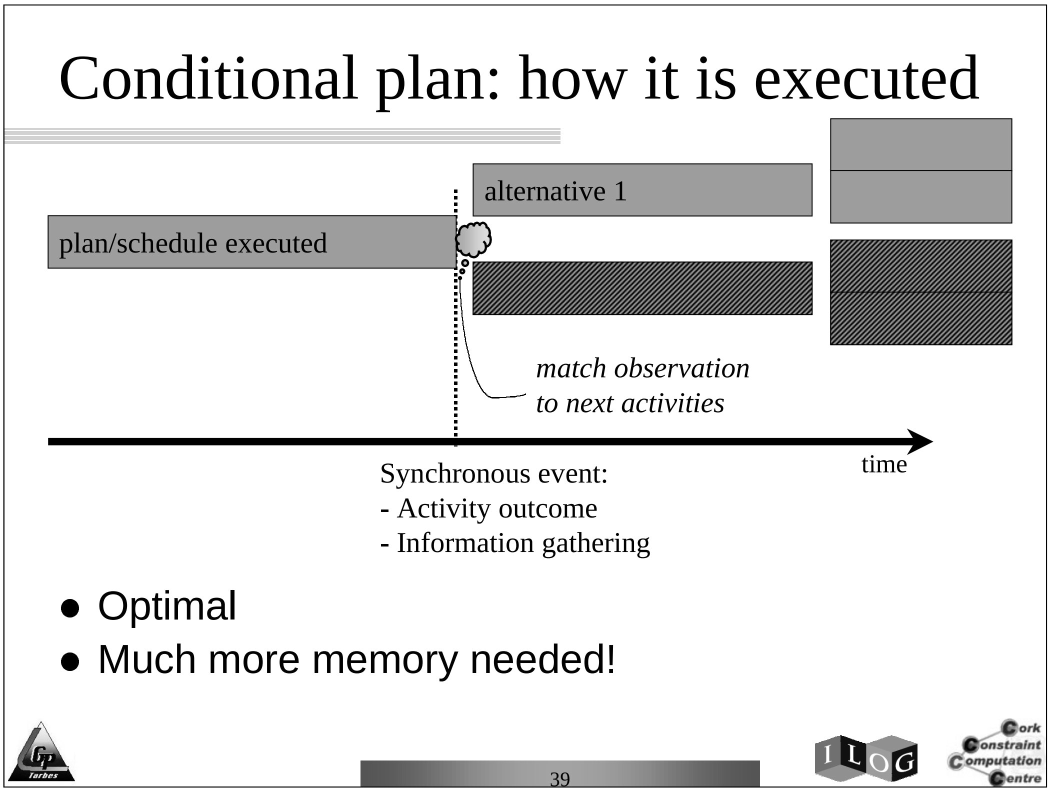
- it only requires **limited memory**.

Legend:

■ activity

▤ current temporal window in which the activity must be set

▨ part of the temporal window squeezed from setting or propagating



Conditional plan: how it is executed

alternative 1

plan/schedule executed

match observation
to next activities

time

Synchronous event:
- Activity outcome
- Information gathering

- Optimal
- Much more memory needed!

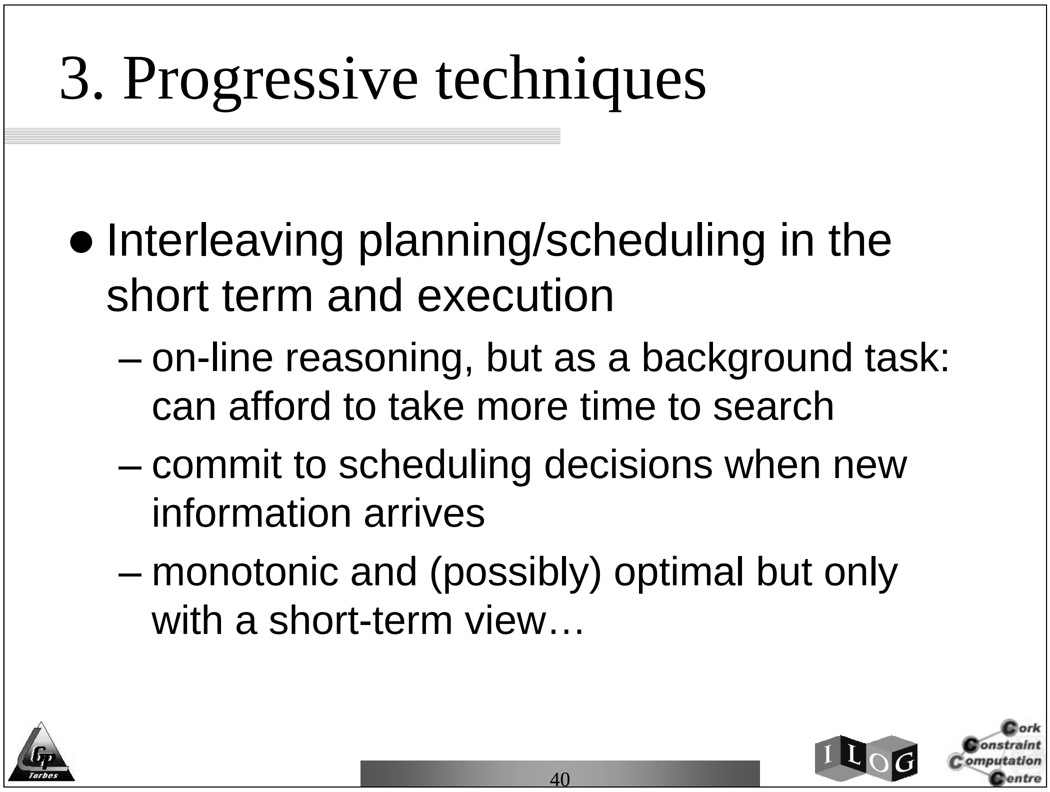
To summarize:

- can be **optimal** since cases are strictly distinguished;

- it is a **monotonic** technique;

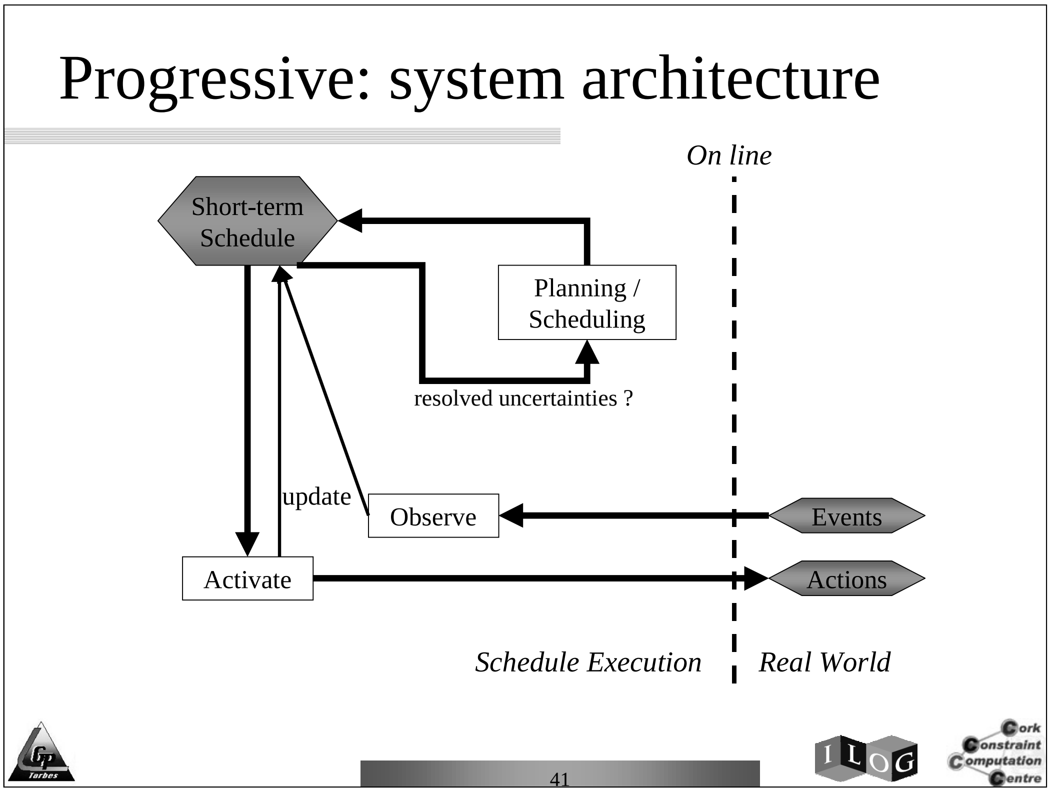
- it copes with **synchronous** events: each time an observation brings the needed information to prune a branch;

- it requires **extended and possibly exploding memory**!



3. Progressive techniques

- Interleaving planning/scheduling in the short term and execution
  - on-line reasoning, but as a background task: can afford to take more time to search
  - commit to scheduling decisions when new information arrives
  - monotonic and (possibly) optimal but only with a short-term view…



Progressive: system architecture

On line

Short-term Schedule

Planning / Scheduling

resolved uncertainties ?

update

Observe

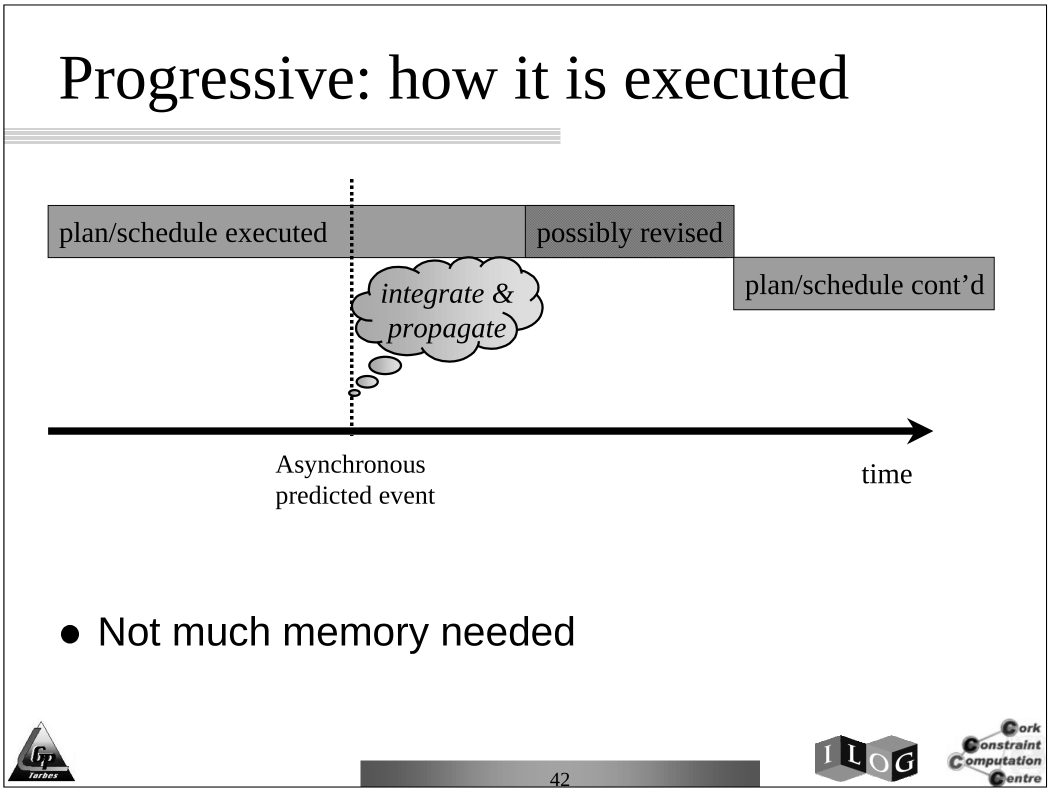
Events

Activate

Actions

Schedule Execution

Real World



Progressive: how it is executed

plan/schedule executed

possibly revised

plan/schedule cont'd

integrate & propagate

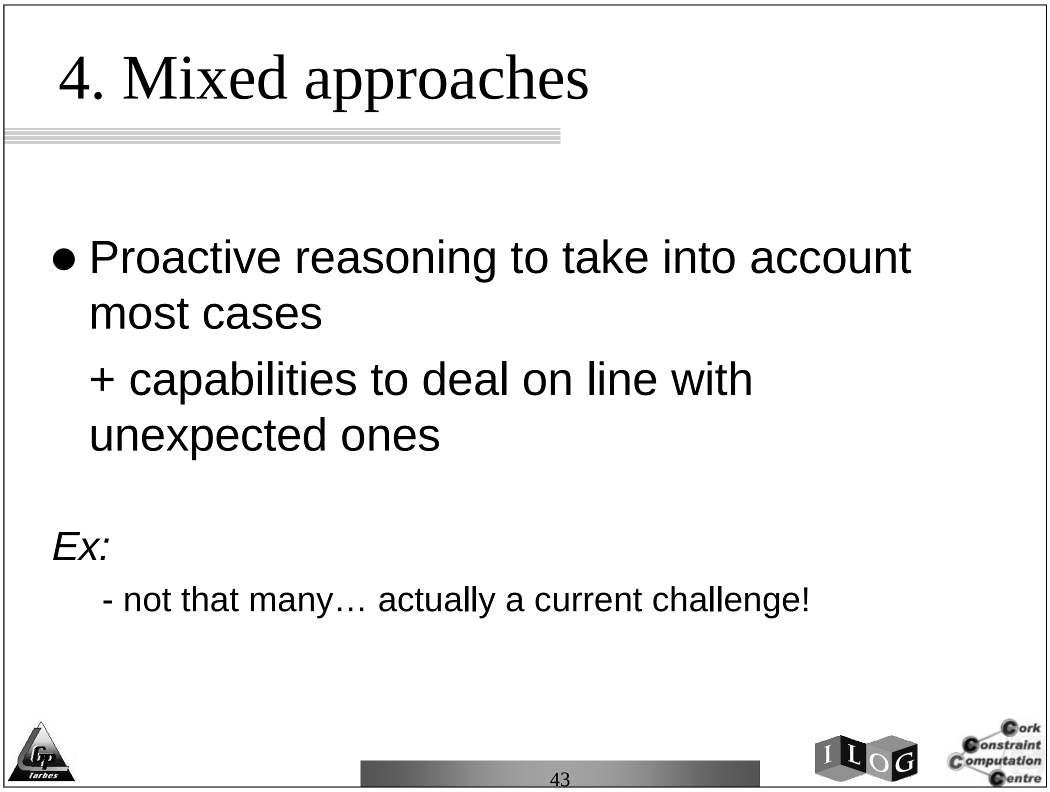
Asynchronous predicted event

time

- Not much memory needed

To summarize:

- need to be **rather quick** but the anticipation leaves time to reason;
- may be **sub-optimal** because it has only a short-term view;
- it is a **monotonic** technique;
- both **synchronous** and **synchronous** events can be accounted for;
- it requires **strictly limited memory** (possibly constant).



4. Mixed approaches

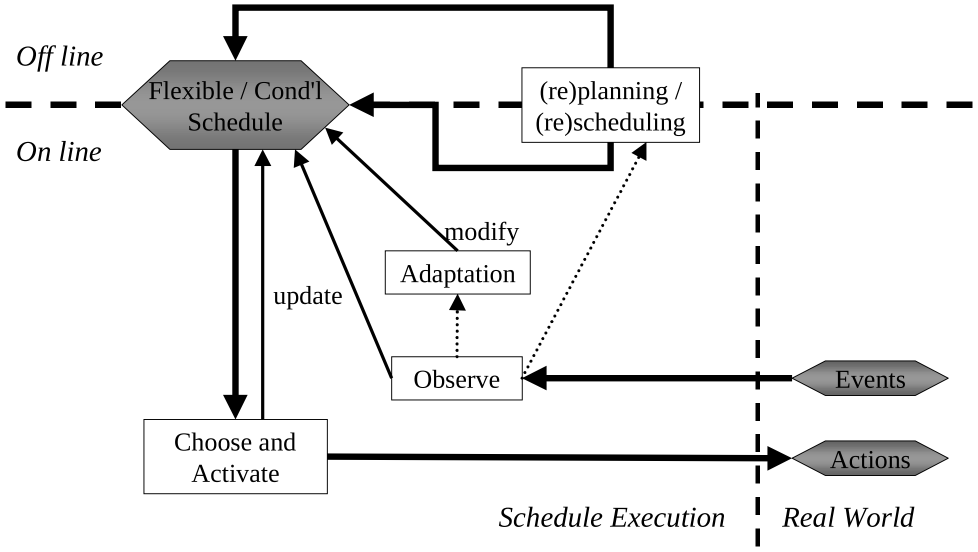
- Proactive reasoning to take into account most cases

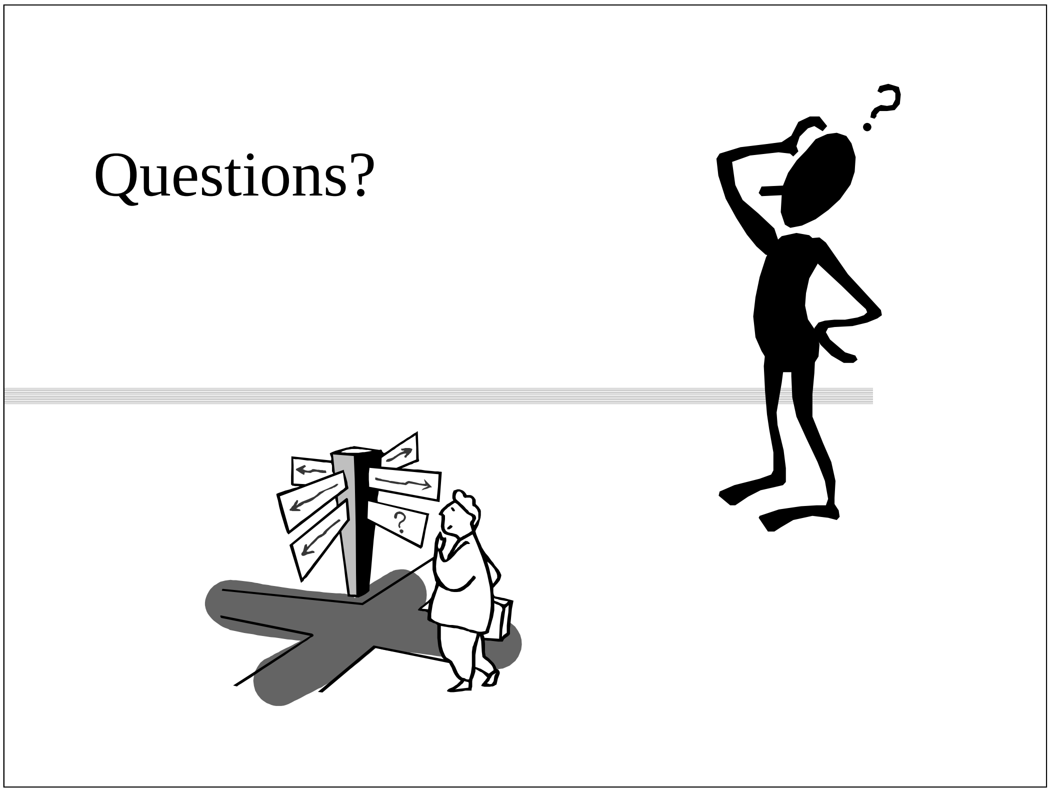
  + capabilities to deal on line with unexpected ones

Ex:

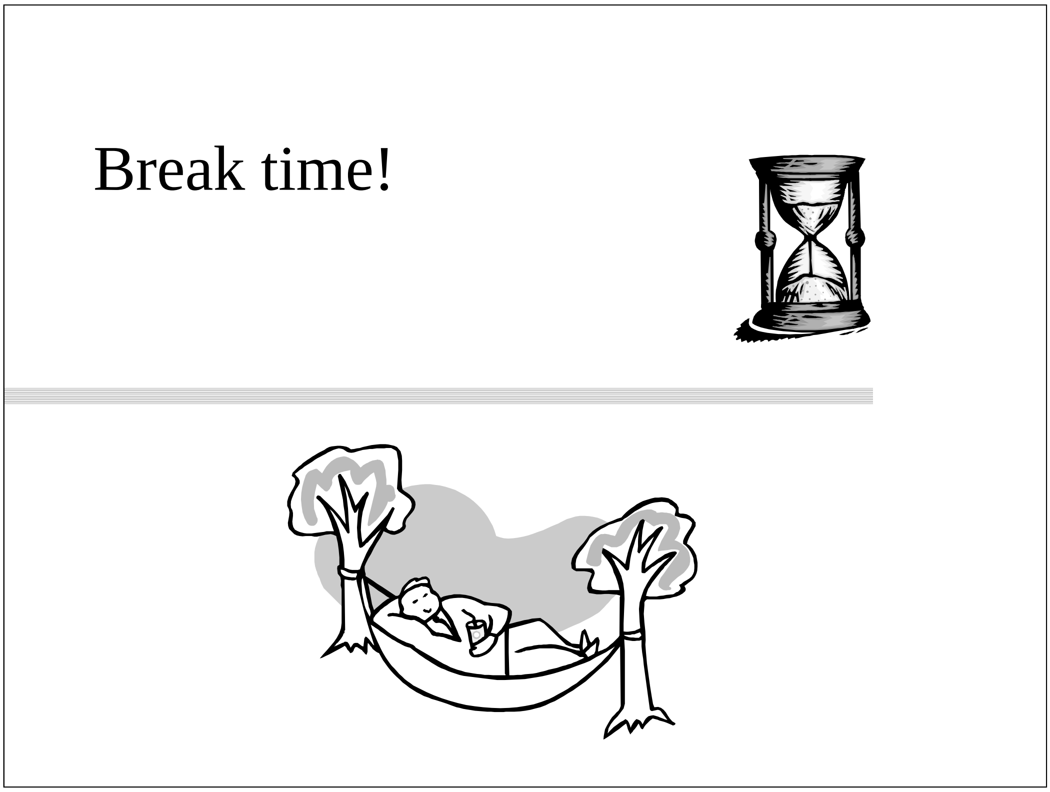
- not that many… actually a current challenge!

Towards a complete architecture...





Questions?

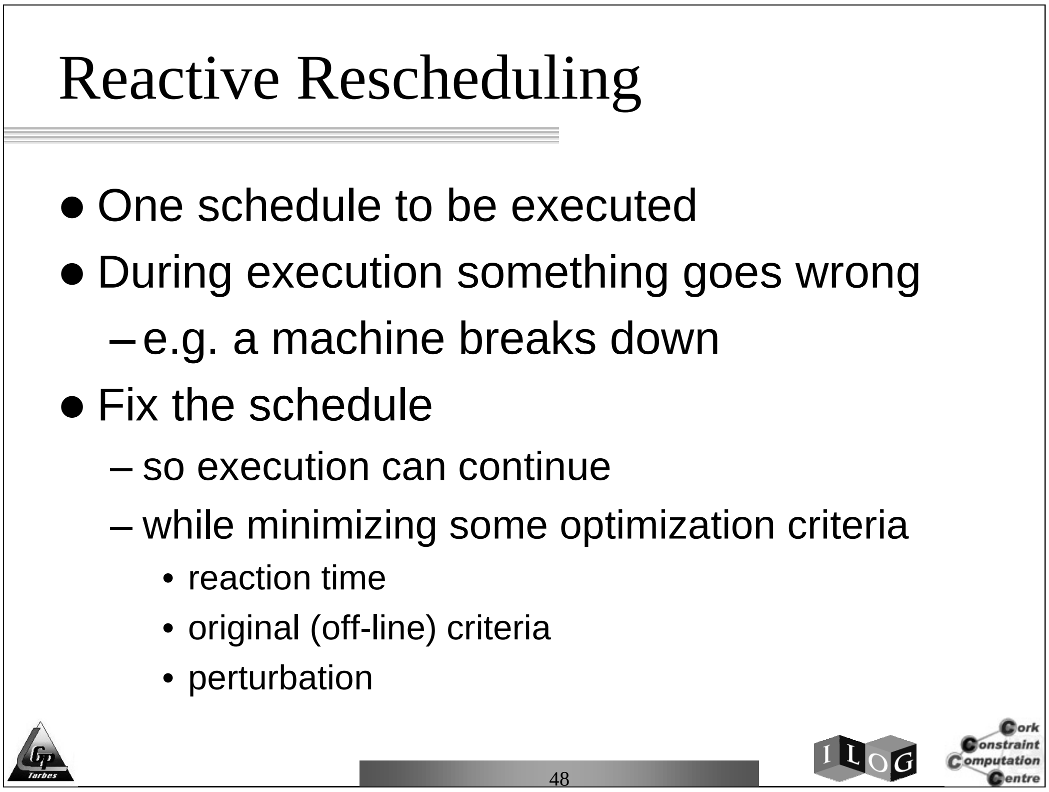


Break time!

Reactive Techniques

Recovery

Replanning/Rescheduling

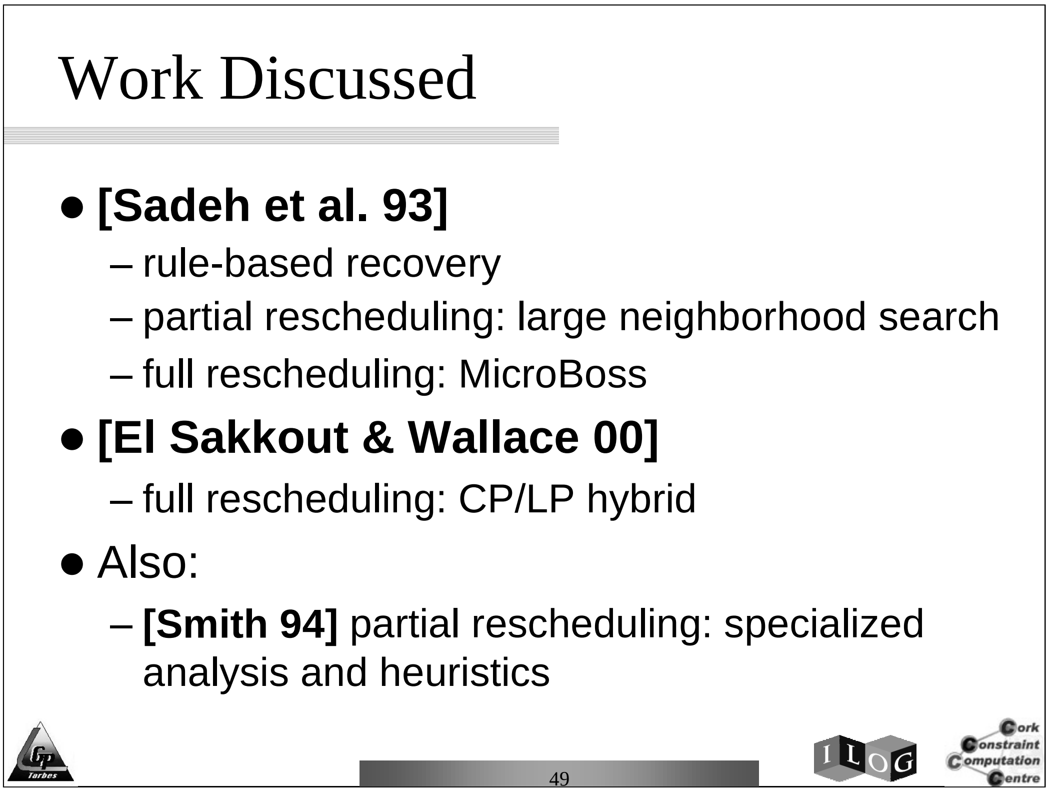


Reactive Rescheduling

- One schedule to be executed
- During execution something goes wrong
  - e.g. a machine breaks down
- Fix the schedule
  - so execution can continue
  - while minimizing some optimization criteria
    - reaction time
    - original (off-line) criteria
    - perturbation

48

This model is distinguished from pure dispatching by the fact that here an off-line schedule exists and some event occurs to make it inconsistent. In pure dispatching, no off-line schedule is created: the sequence in which activities are executed is determined entirely at execution time.



Work Discussed

- **[Sadeh et al. 93]**
  - – rule-based recovery
  - – partial rescheduling: large neighborhood search
  - – full rescheduling: MicroBoss
- **[El Sakkout & Wallace 00]**
  - – full rescheduling: CP/LP hybrid
- Also:
  - – **[Smith 94]** partial rescheduling: specialized analysis and heuristics

1&2. **[Sadeh et al. 93]** Sadeh, N., Otsuka, S. and Schelback, R. *Predictive and reactive scheduling with the MicroBoss production scheduling system*. In Proceedings of the IJCAI'93 Workshop on Production Planning, Scheduling, and Control. 1993.

3. **[Smith 94]** Smith, S.F. *OPIS: A methodology and architecture for reactive scheduling* in Intelligent Scheduling, Morgan Kaufman, 1994.

4. **[El Sakkout & Wallace 00]** El Sakkout, H. and Wallace, M. *Probe backtrack search for minimal perturbation in dynamic scheduling,* CONSTRAINTS, 5(4), 2000.

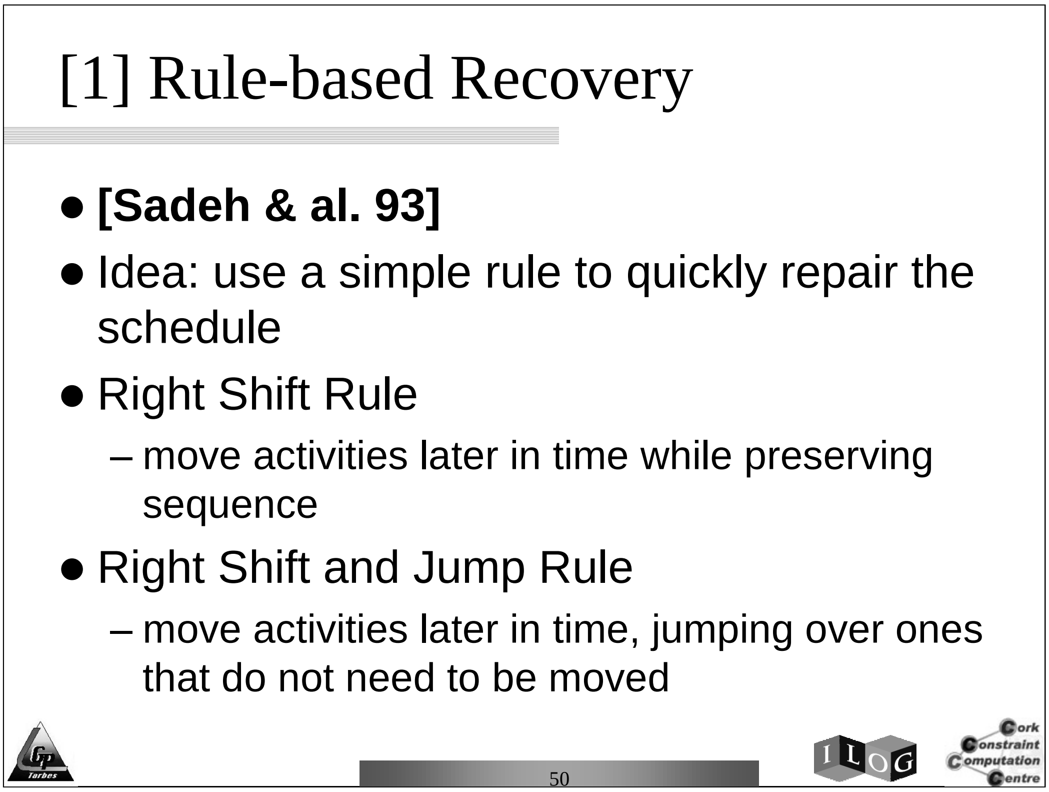
See Also:

[Ow et al. 88] Ow, P.S., Smith, S.F., and Thiriez, A. *Reactive Plan Revision,* AAAI’88, pp. 77-82, 1988.

Large neighborhood local search: a general local search technique where a subset of decisions to “undo” is identified. Then a constructive search technique is used to resolve.

CP - Constraint Programming

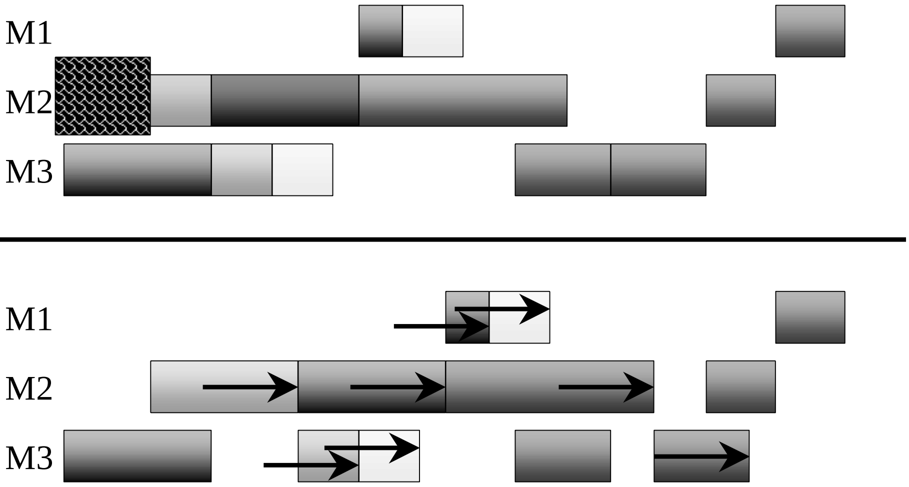
LP - Linear Programming



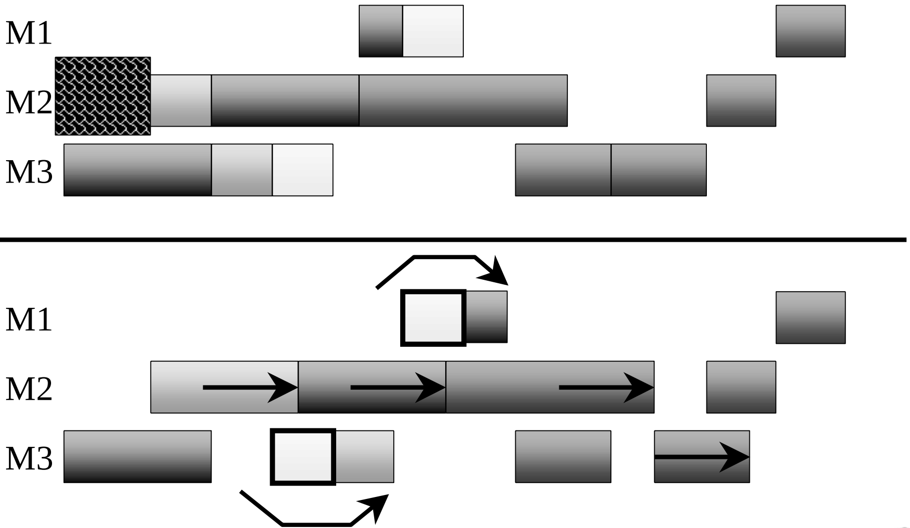
[1] Rule-based Recovery

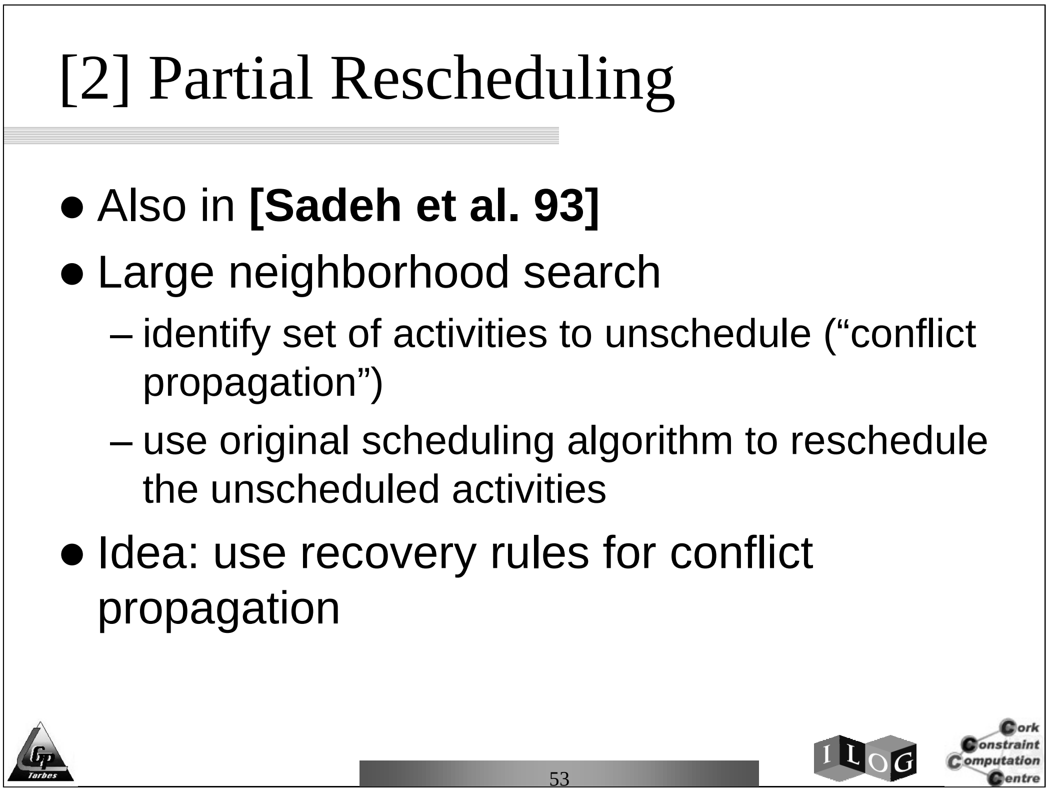
- **[Sadeh & al. 93]**
- Idea: use a simple rule to quickly repair the schedule
- Right Shift Rule
  - move activities later in time while preserving sequence
- Right Shift and Jump Rule
  - move activities later in time, jumping over ones that do not need to be moved

[1] Right Shift (RSh)



[1] Right Shift and Jump (RShJ)

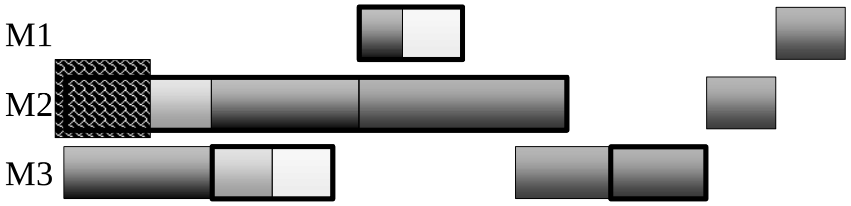


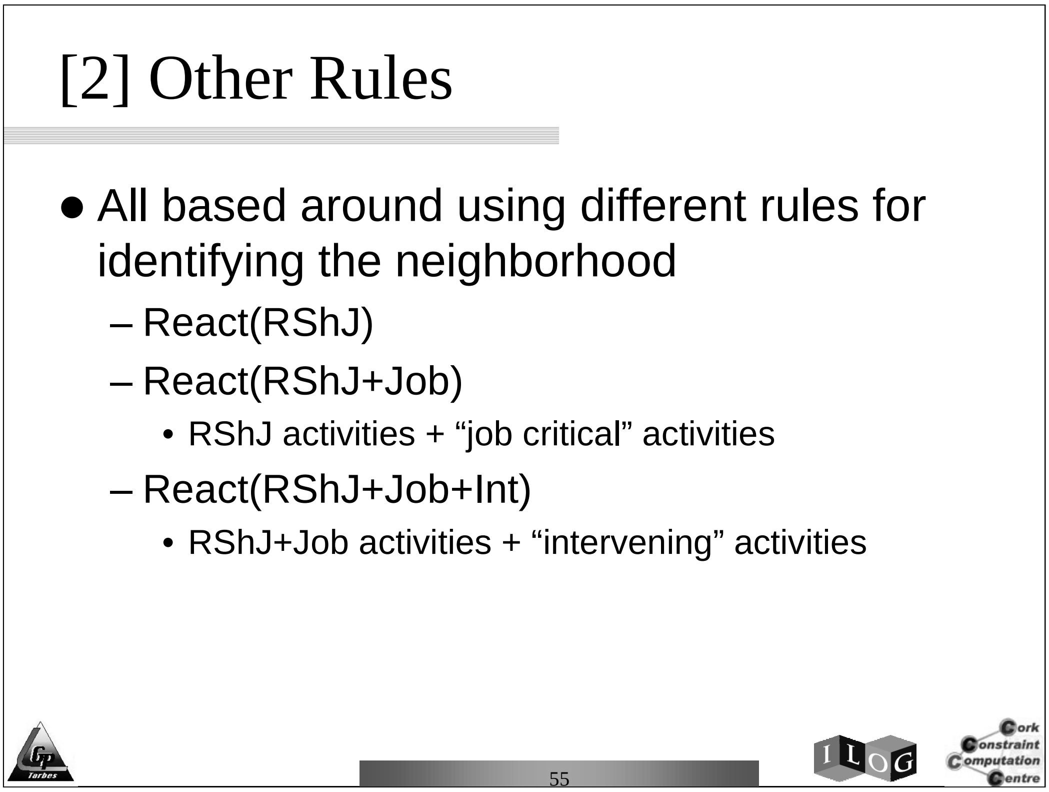


[2] Partial Rescheduling

- Also in **[Sadeh et al. 93]**
- Large neighborhood search
  - – identify set of activities to unschedule (“conflict propagation”)
  - – use original scheduling algorithm to reschedule the unscheduled activities
- Idea: use recovery rules for conflict propagation

[2] React(RSh)





[2] Other Rules

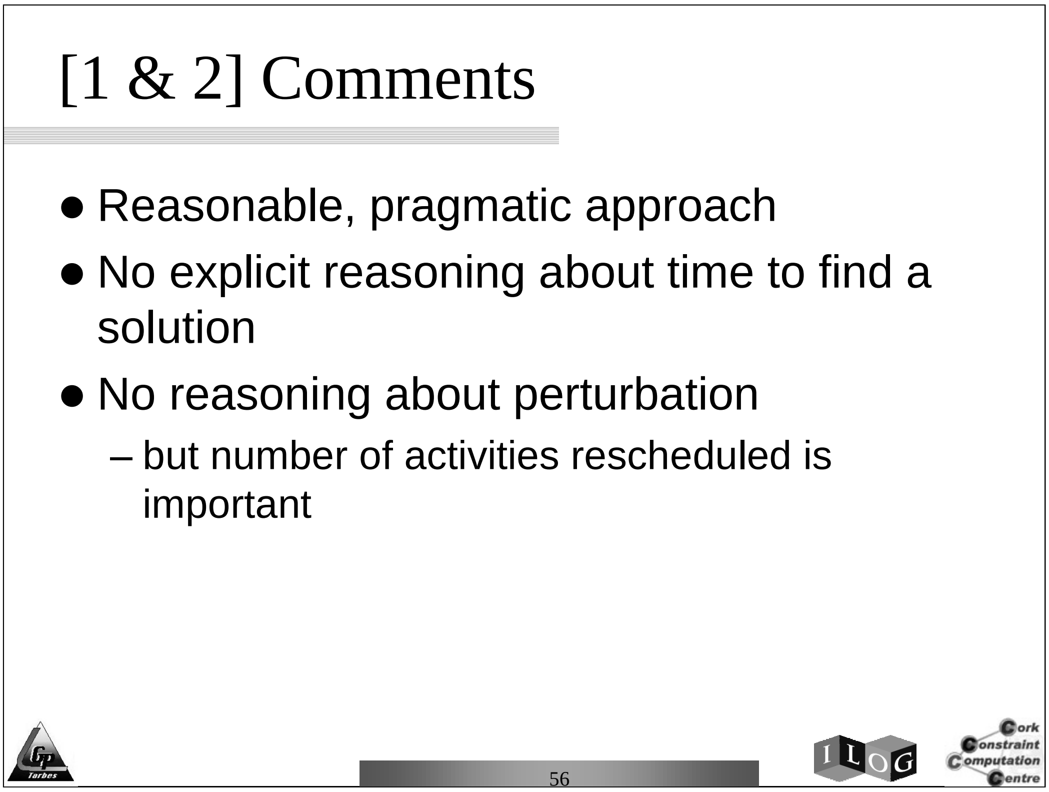
- All based around using different rules for identifying the neighborhood
  - React(RShJ)
  - React(RShJ+Job)
    - RShJ activities + “job critical” activities
  - React(RShJ+Job+Int)
    - RShJ+Job activities + “intervening” activities

Job critical activities:

- if the second last activity in a job has been unscheduled, unschedule the last activity
- if the second activity in a job has been unscheduled, unschedule the first activity

Intervening activities:

- if two activities surrounding activity A on resource R have been unscheduled, unschedule A
- if two activities surrounding activity A in job J have been unscheduled, unschedule A



[1 & 2] Comments

- Reasonable, pragmatic approach
- No explicit reasoning about time to find a solution
- No reasoning about perturbation
  - – but number of activities rescheduled is important

Pragmatic: very clear that this is a way that one can build a system
Also identifies a combined approach (no experiments):

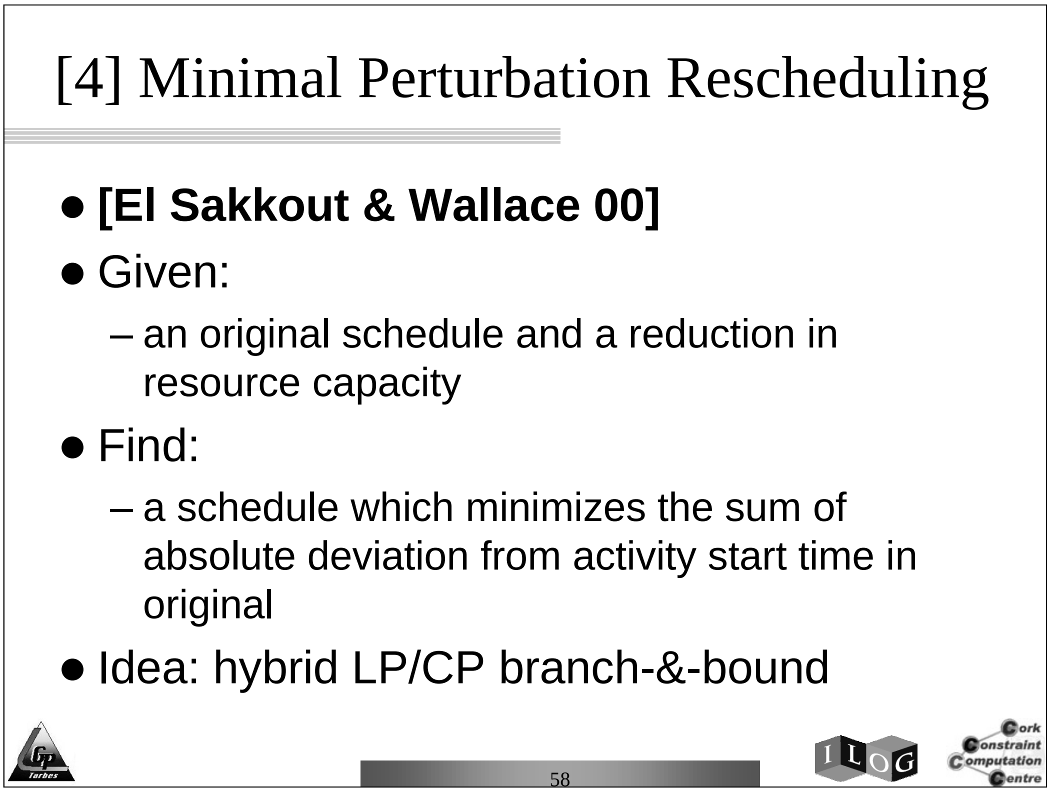
- Control level: small changes use rule-based recovery
- Scheduling level: larger disruptions use partial rescheduling

Does not make any contributions in answering the questions:

- when to use rules (“control level”) and when to use partial rescheduling (”scheduling level”)?
- are there principled ways for identifying the neighborhood?

[3] Partial Rescheduling

- **[Smith 94]**
- OPIS
  - full scheduling system based on repeatedly reacting to events
  - predates [Sadeh 93]
  - more sophisticated (AI-ish) mechanism for analysis of conflicts
  - strong performance in real applications



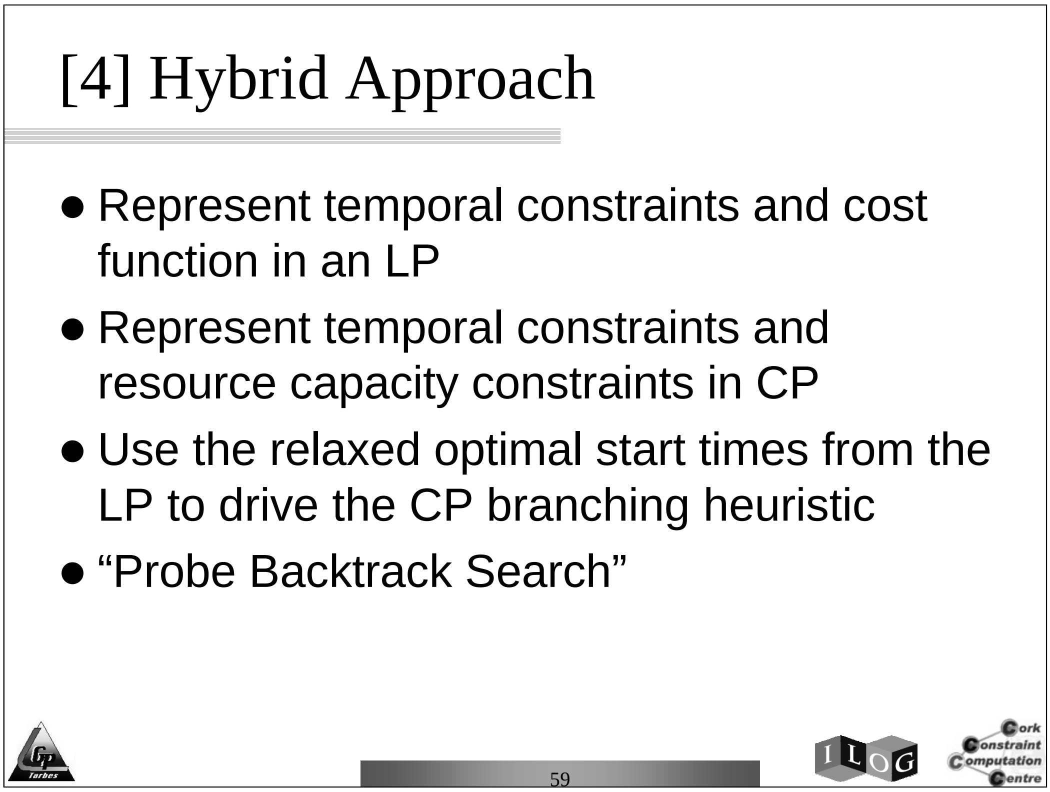
[4] Minimal Perturbation Rescheduling

- **[El Sakkout & Wallace 00]**
- Given:
  - an original schedule and a reduction in resource capacity
- Find:
  - a schedule which minimizes the sum of absolute deviation from activity start time in original
- Idea: hybrid LP/CP branch-&-bound

Recall the approaches to reactive scheduling:

- rule-based recovery
- partial rescheduling
- complete rescheduling

This is an extreme approach.

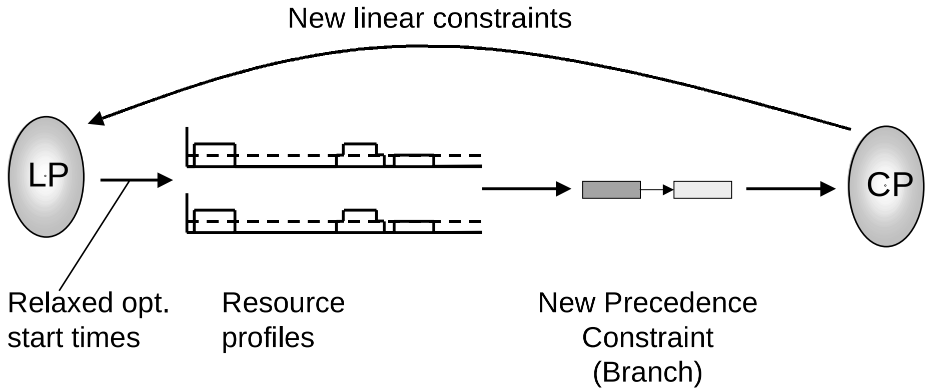


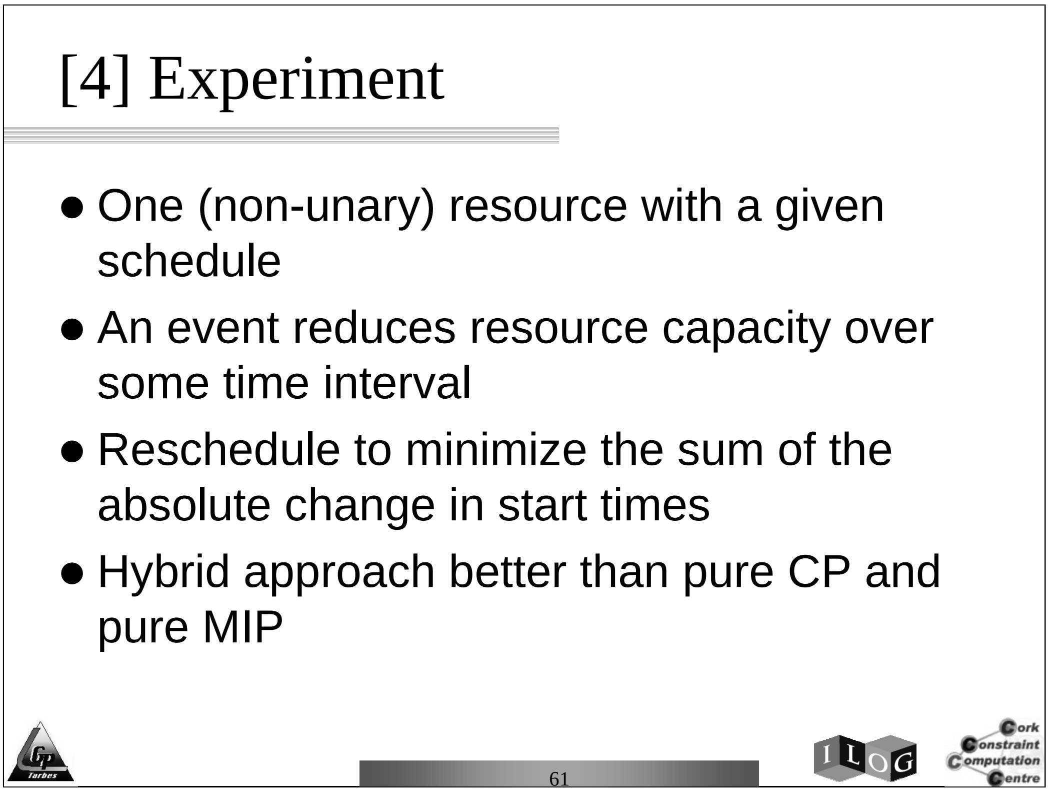
[4] Hybrid Approach

- Represent temporal constraints and cost function in an LP
- Represent temporal constraints and resource capacity constraints in CP
- Use the relaxed optimal start times from the LP to drive the CP branching heuristic
- “Probe Backtrack Search”

LP – linear programming
CP – constraint programming

[4] Probe Backtrack Search



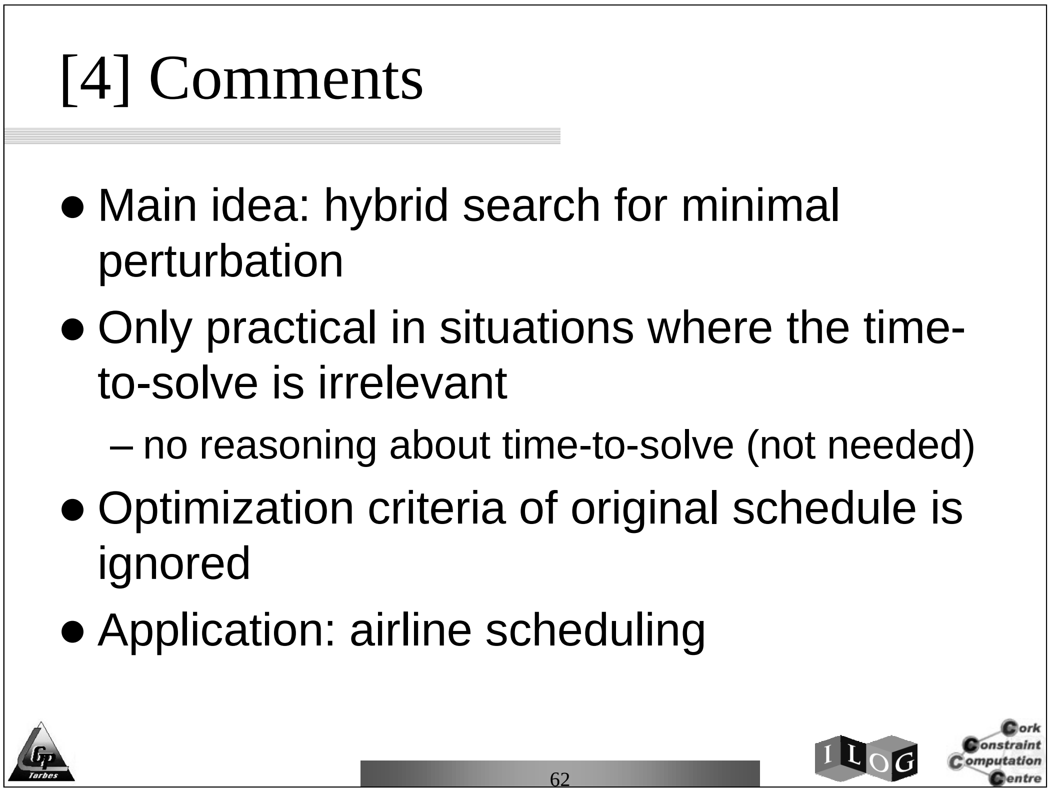


[4] Experiment

- One (non-unary) resource with a given schedule
- An event reduces resource capacity over some time interval
- Reschedule to minimize the sum of the absolute change in start times
- Hybrid approach better than pure CP and pure MIP

61

Non-unary resource – resource can process more than one activity at a time. Often called a *discrete* resource.

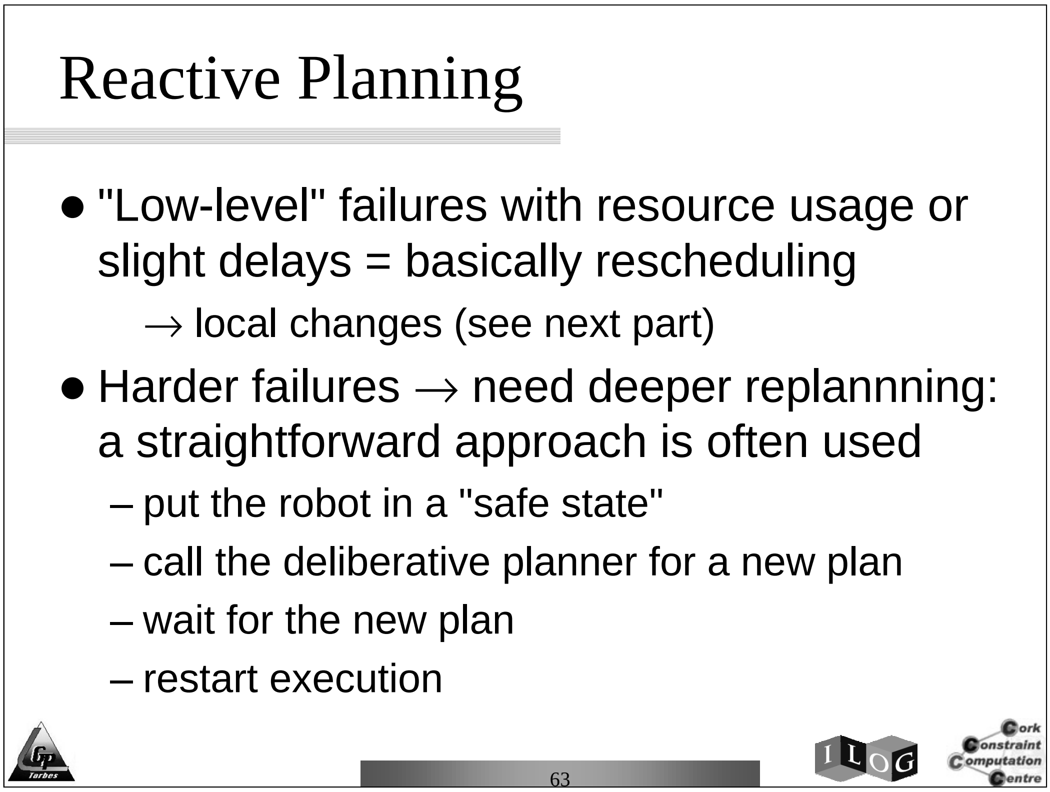


[4] Comments

- Main idea: hybrid search for minimal perturbation
- Only practical in situations where the time-to-solve is irrelevant
  - no reasoning about time-to-solve (not needed)
- Optimization criteria of original schedule is ignored
- Application: airline scheduling

Minimal perturbation rescheduling is useful when:

- activity durations are long compared to the solve time
- information about a breakdown is known in advance
- it is very expensive to change already scheduled activities



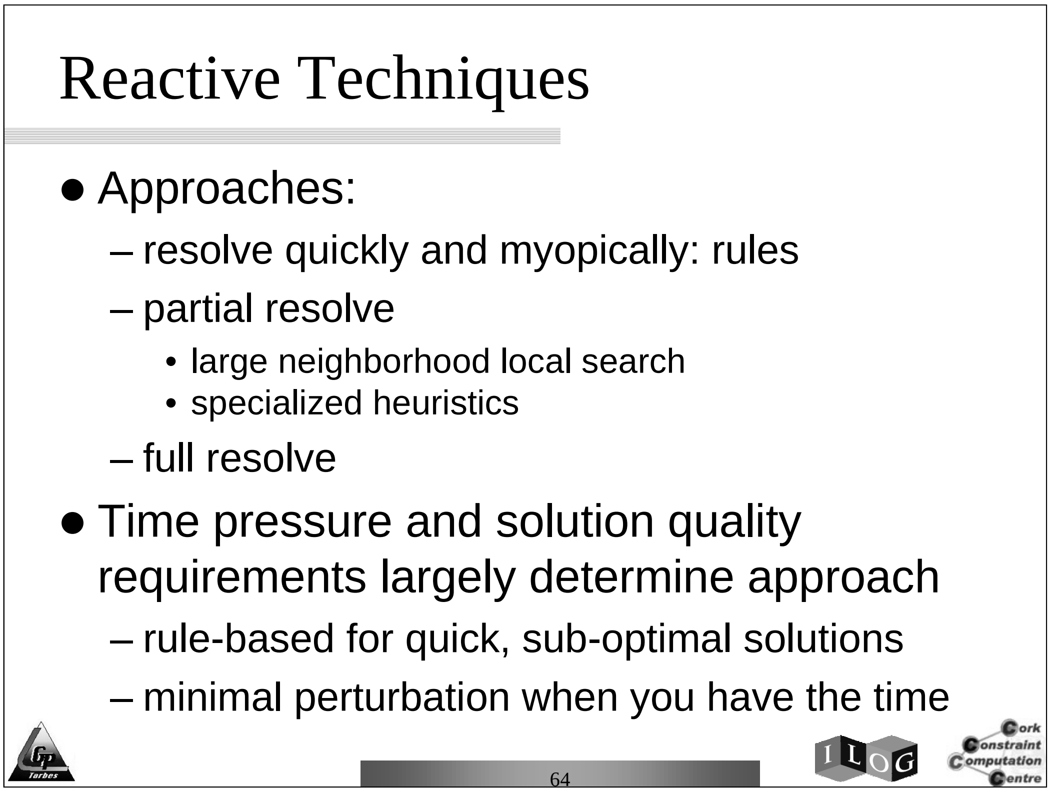
Reactive Planning

- "Low-level" failures with resource usage or slight delays = basically rescheduling

  → local changes (see next part)

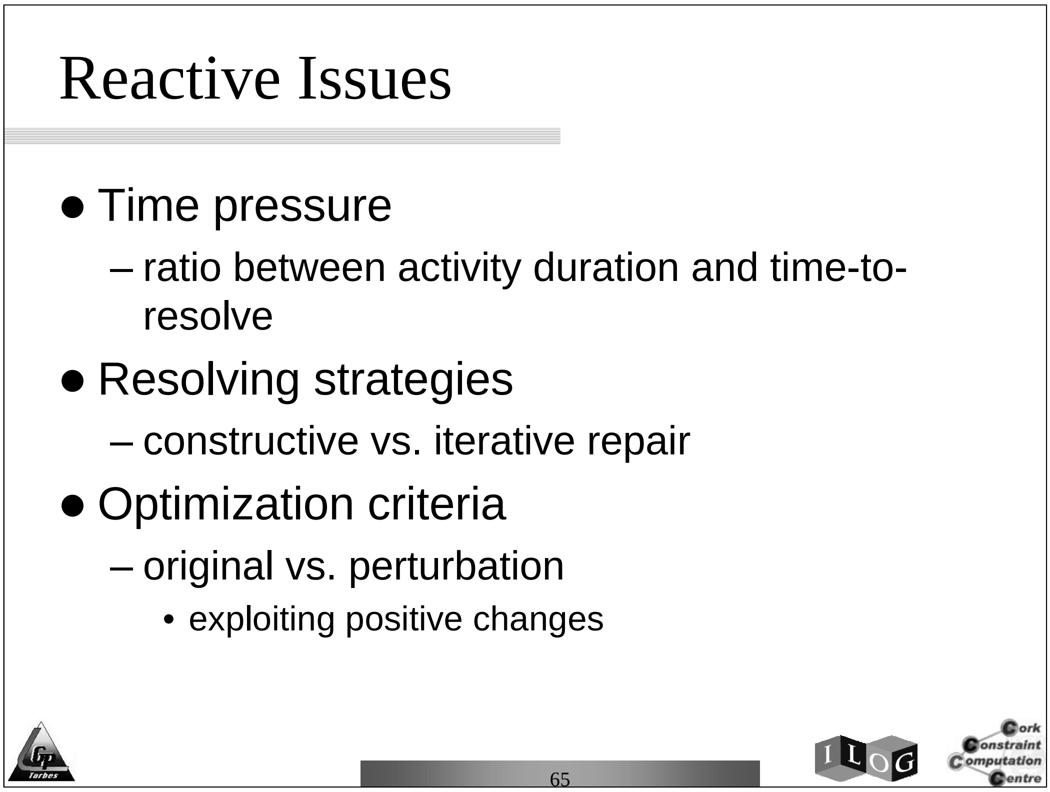
- Harder failures → need deeper replannning: a straightforward approach is often used
  - put the robot in a "safe state"
  - call the deliberative planner for a new plan
  - wait for the new plan
  - restart execution

We will consider examples of planning with reactive capabilities in the "mixed approaches" section.



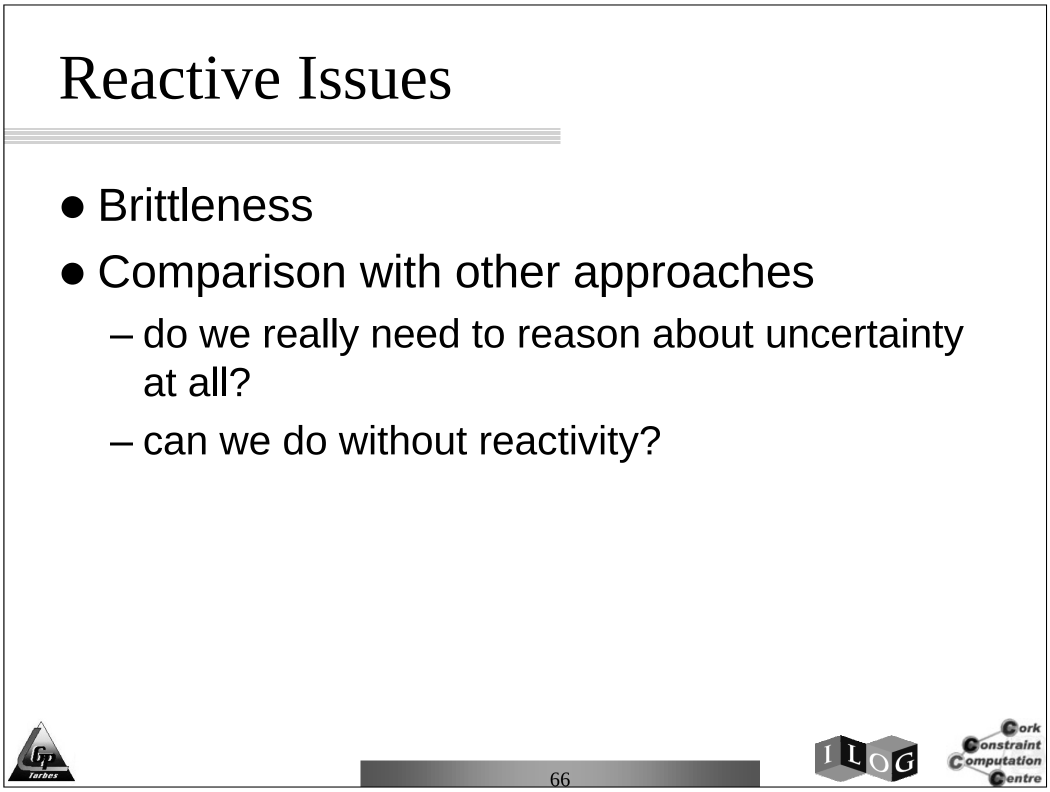
Reactive Techniques

- Approaches:
  - resolve quickly and myopically: rules
  - partial resolve
    - large neighborhood local search
    - specialized heuristics
  - full resolve
- Time pressure and solution quality requirements largely determine approach
  - rule-based for quick, sub-optimal solutions
  - minimal perturbation when you have the time



Reactive Issues

- Time pressure
  - ratio between activity duration and time-to-resolve
- Resolving strategies
  - constructive vs. iterative repair
- Optimization criteria
  - original vs. perturbation
    - exploiting positive changes



Reactive Issues

- Brittleness
- Comparison with other approaches
  - do we really need to reason about uncertainty at all?
  - can we do without reactivity?

66

Brittleness: after a number of reactive repairs, does the solution become increasingly brittle? That is, can a small disruption have disproportionate effects on the schedule?

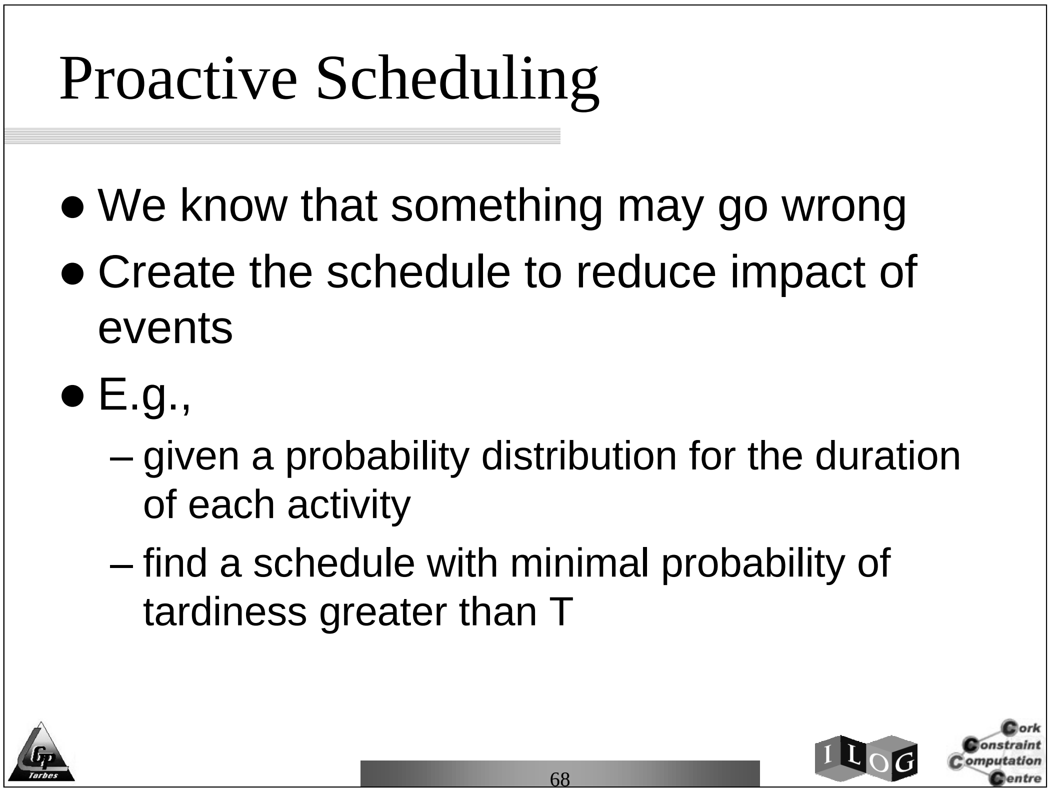
Learning to react …

Proactive Off-line Techniques

Maximal coverage

Flexible models

Conditional models

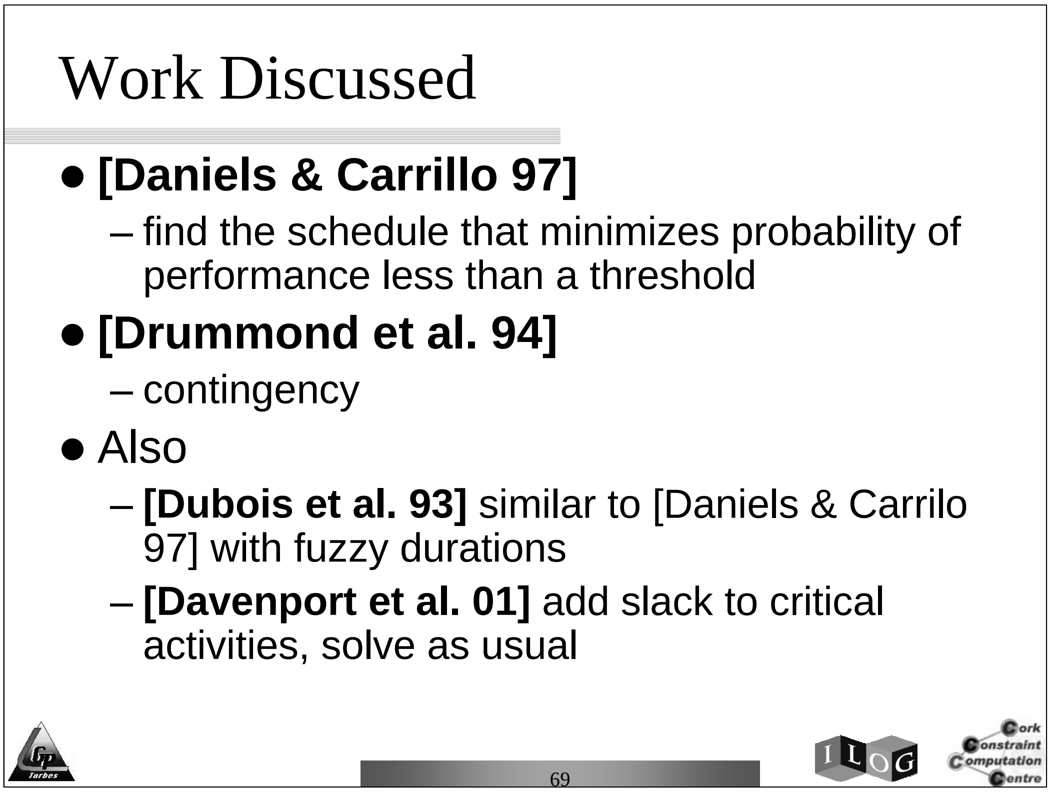


Proactive Scheduling

- We know that something may go wrong
- Create the schedule to reduce impact of events
- E.g.,
  - given a probability distribution for the duration of each activity
  - find a schedule with minimal probability of tardiness greater than T

68

This model does not pay any attention to what will happen at execution time to handle unexpected events. It is purely off-line.



Work Discussed

- **[Daniels & Carrillo 97]**
  - find the schedule that minimizes probability of performance less than a threshold
- **[Drummond et al. 94]**
  - contingency
- Also
  - **[Dubois et al. 93]** similar to [Daniels & Carrilo 97] with fuzzy durations
  - **[Davenport et al. 01]** add slack to critical activities, solve as usual

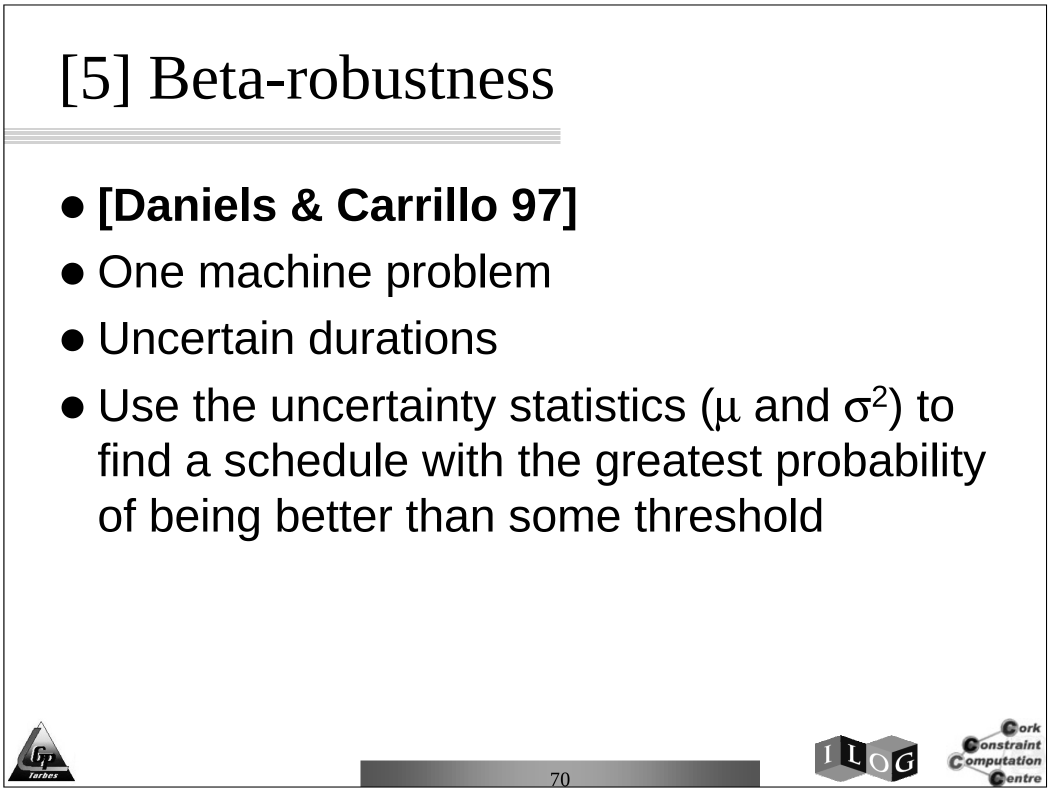
69

5. **[Daniels & Carrillo 97]** Daniels, R.L. and Carrillo, J.E. *β-Robust scheduling for single-machine systems with uncertain processing times*, IIE Transactions, 29, 977-985, 1997.

6. **[Dubois 93]** Dubois, D., Fargier, F. and Prade, H. *The use of fuzzy constraints in job-shop scheduling*, Proceedings of the IJCAI-93 Workshop on Knowledge-Based Planning, Scheduling and Control, Chambéry, 1993.

7. **[Davenport et al. 01]** Davenport, A.J., Gefflot, C., and Beck, J.C. *Slack-based techniques for building robust schedules*, Proceedings of the Sixth European Conference on Planning (ECP-01), 2001.

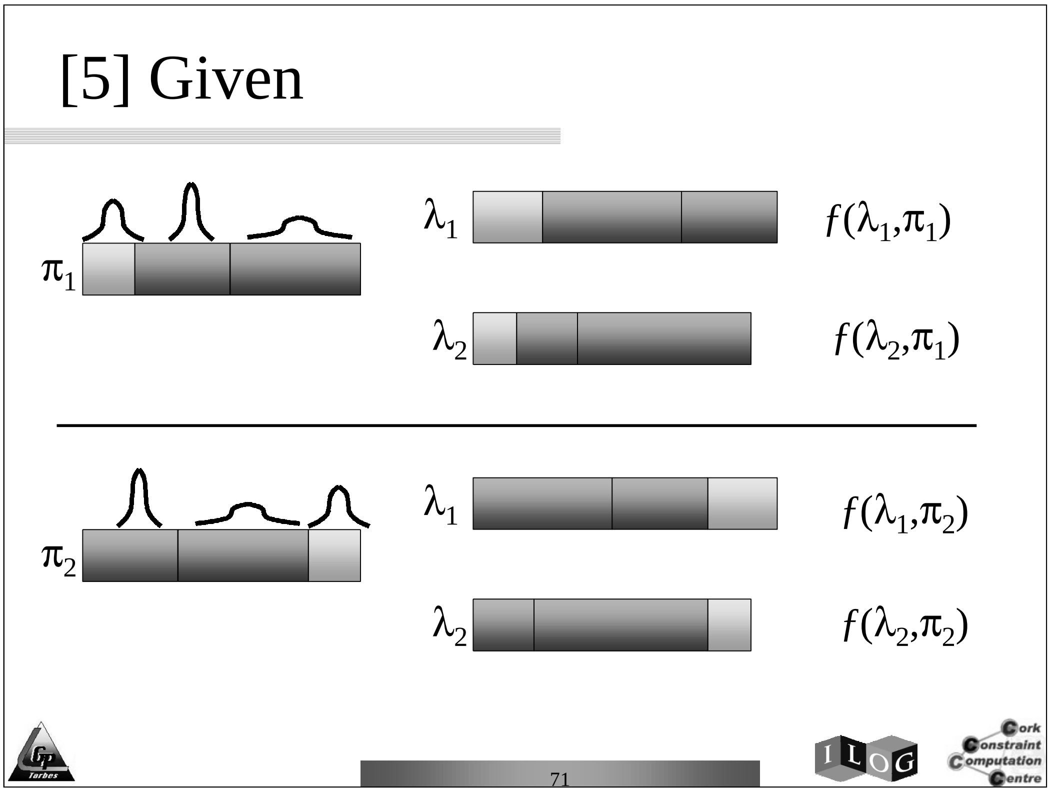
8. **[Drummond et al. 94]** Drummond, M., Bresina, J., & Swanson, K. *Just-in-case scheduling*, Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), 1994.



[5] Beta-robustness

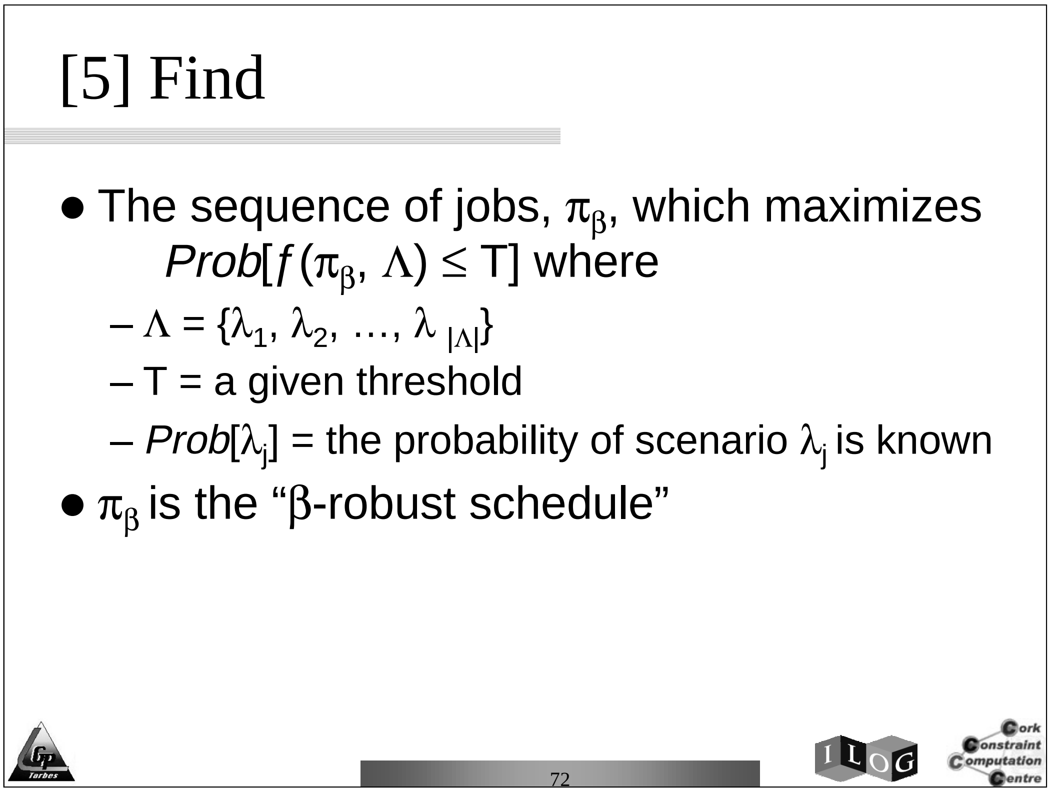
- **[Daniels & Carrillo 97]**
- One machine problem
- Uncertain durations
- Use the uncertainty statistics (μ and σ^2) to find a schedule with the greatest probability of being better than some threshold

70



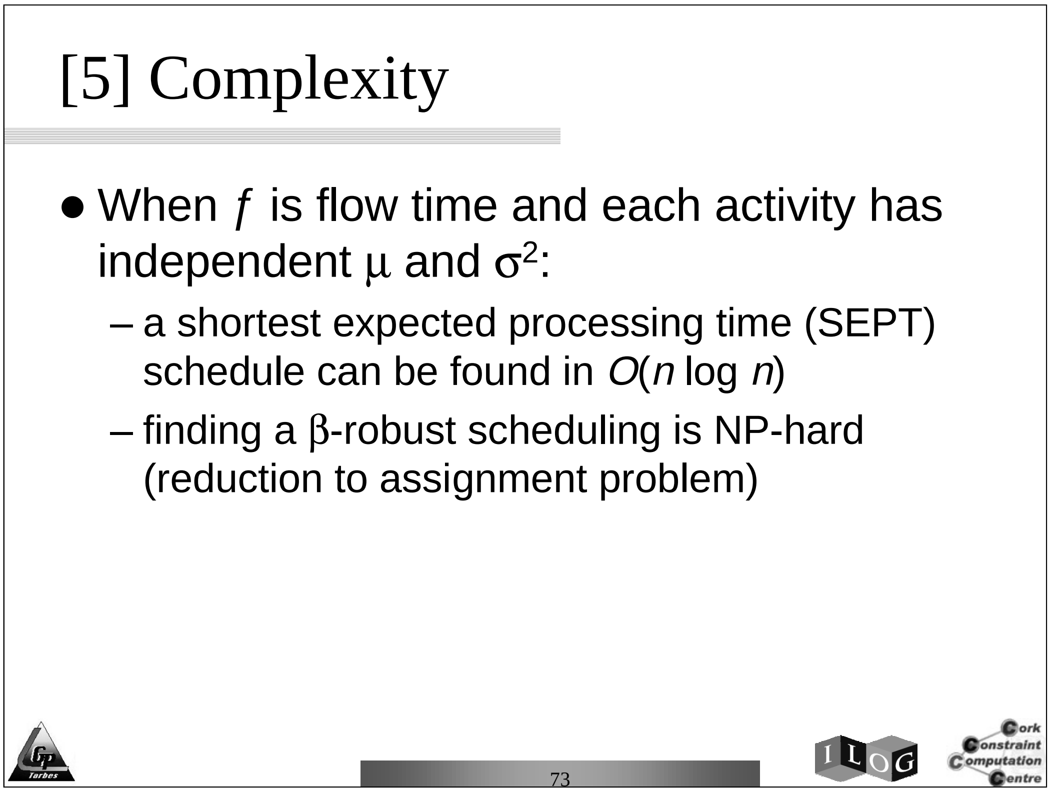
[5] Given

π_1
λ_1
$f(\lambda_1,\pi_1)$
λ_2
$f(\lambda_2,\pi_1)$
π_2
λ_1
$f(\lambda_1,\pi_2)$
λ_2
$f(\lambda_2,\pi_2)$



[5] Find

- The sequence of jobs, π_β, which maximizes *Prob*[$f(\pi_\beta, \Lambda) \leq$ T] where
  - – $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_{|\Lambda|}\}$
  - – T = a given threshold
  - – *Prob*[λ_j] = the probability of scenario λ_j is known
- π_β is the “β-robust schedule”

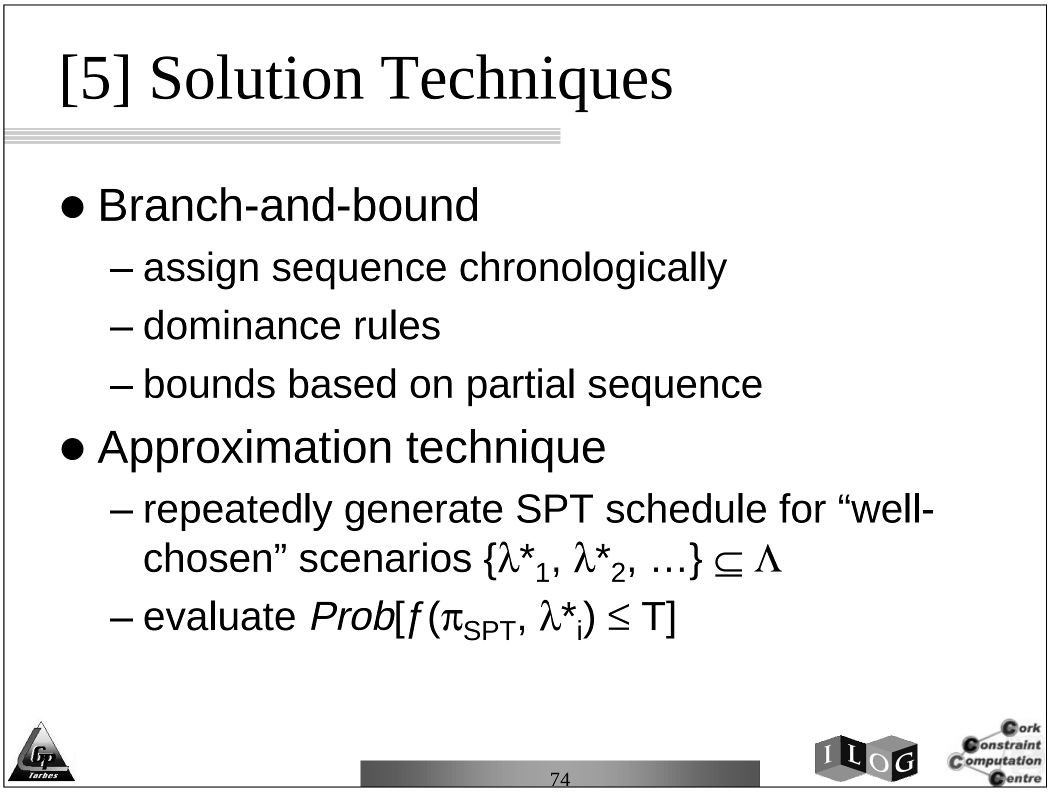


[5] Complexity

- When *f* is flow time and each activity has independent μ and σ^2:
  - a shortest expected processing time (SEPT) schedule can be found in $O(n \log n)$
  - finding a β-robust scheduling is NP-hard (reduction to assignment problem)

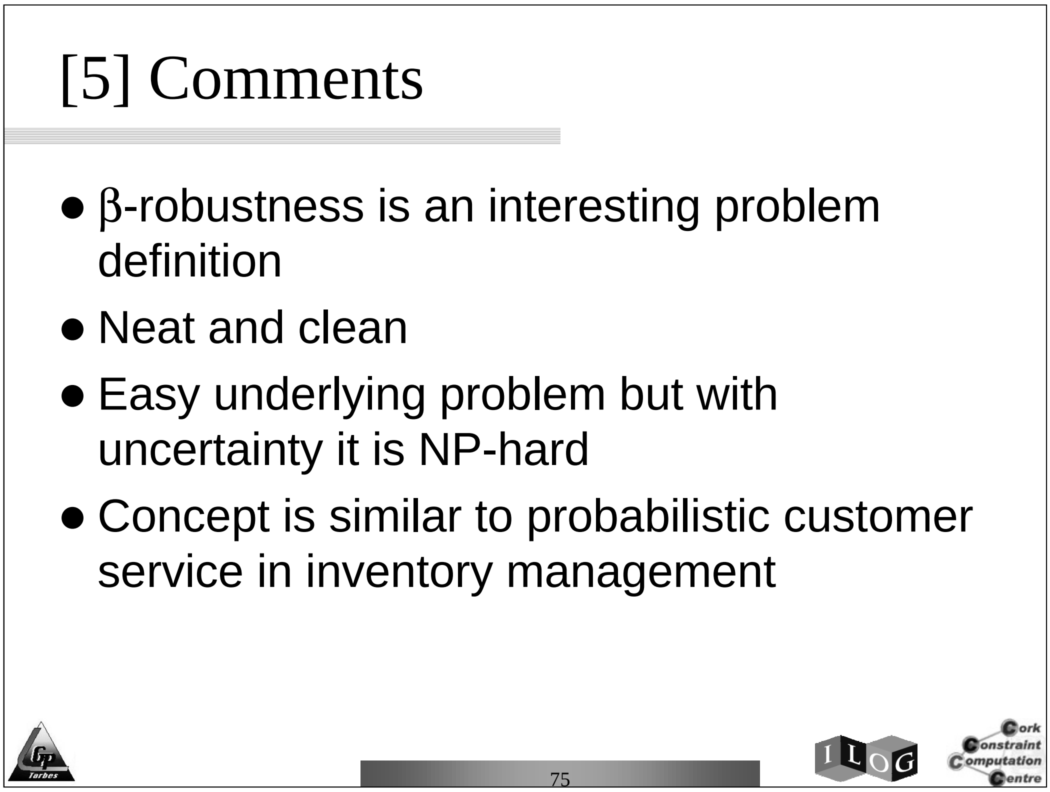
flow time: the sum (over all activities) of the interval of time between the release time of the activity and when its execution is finished.
The release of an activity is the earliest time at which it can be scheduled.
Here, the release time for all activities is 0.
Example: two activities, A and B, with durations of 10 and 100 respectively.
The release time for each is 0.
Flow time = (endtime(A) – 0) + (endtime(B) – 0)

- A → B: flow time = (10 – 0) + (110 – 0) = 120
- B → A: flow time = (110 – 0) + (100 – 0) = 210



[5] Solution Techniques

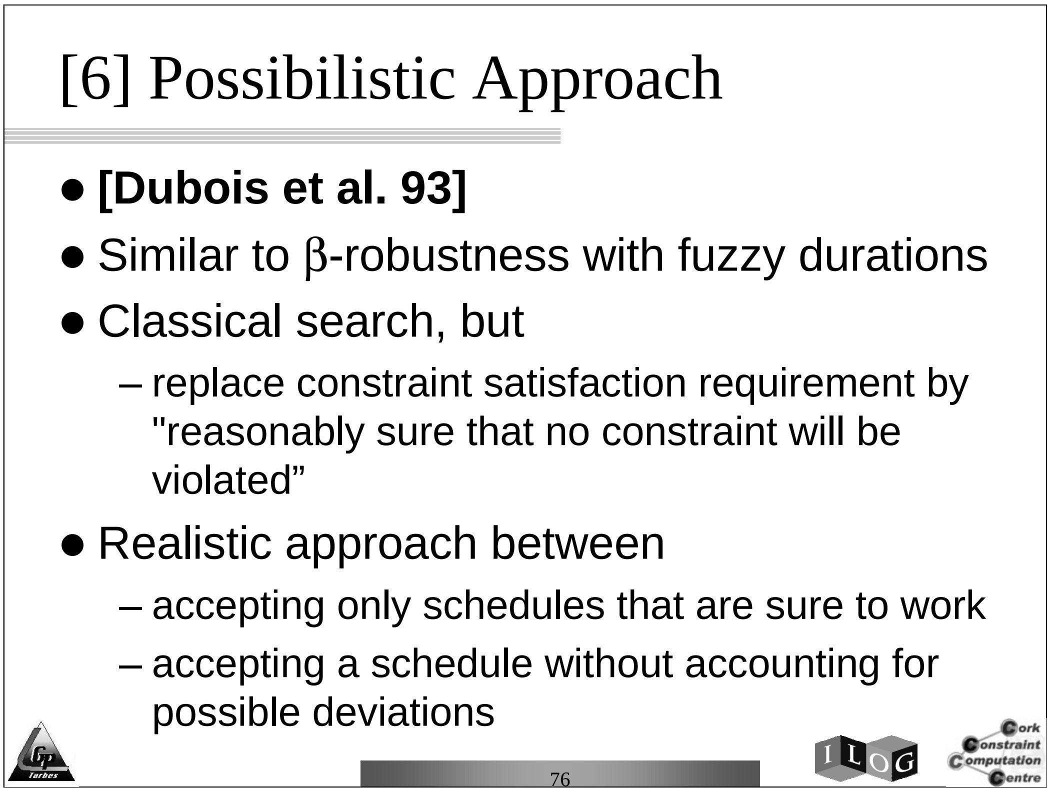
- Branch-and-bound
  - – assign sequence chronologically
  - – dominance rules
  - – bounds based on partial sequence
- Approximation technique
  - – repeatedly generate SPT schedule for “well-chosen” scenarios $\{\lambda^*_1, \lambda^*_2, \ldots\} \subseteq \Lambda$
  - – evaluate $\mathit{Prob}[f(\pi_{SPT}, \lambda^*_i) \leq T]$



[5] Comments

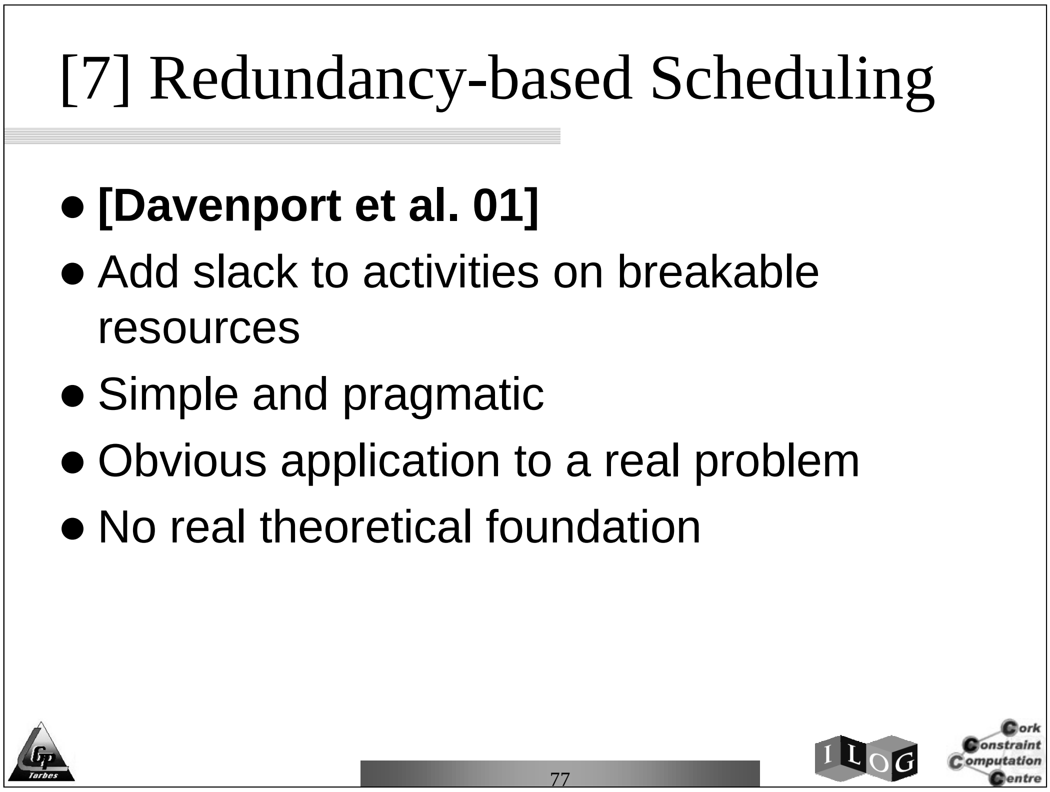
- β-robustness is an interesting problem definition
- Neat and clean
- Easy underlying problem but with uncertainty it is NP-hard
- Concept is similar to probabilistic customer service in inventory management

Probabilistic customer service allocates inventory so that there are probabilistic guarantees on achieving full customer service. For example, a 95% customer service level means that 95% of the time all customer orders will be met from the stored inventory.



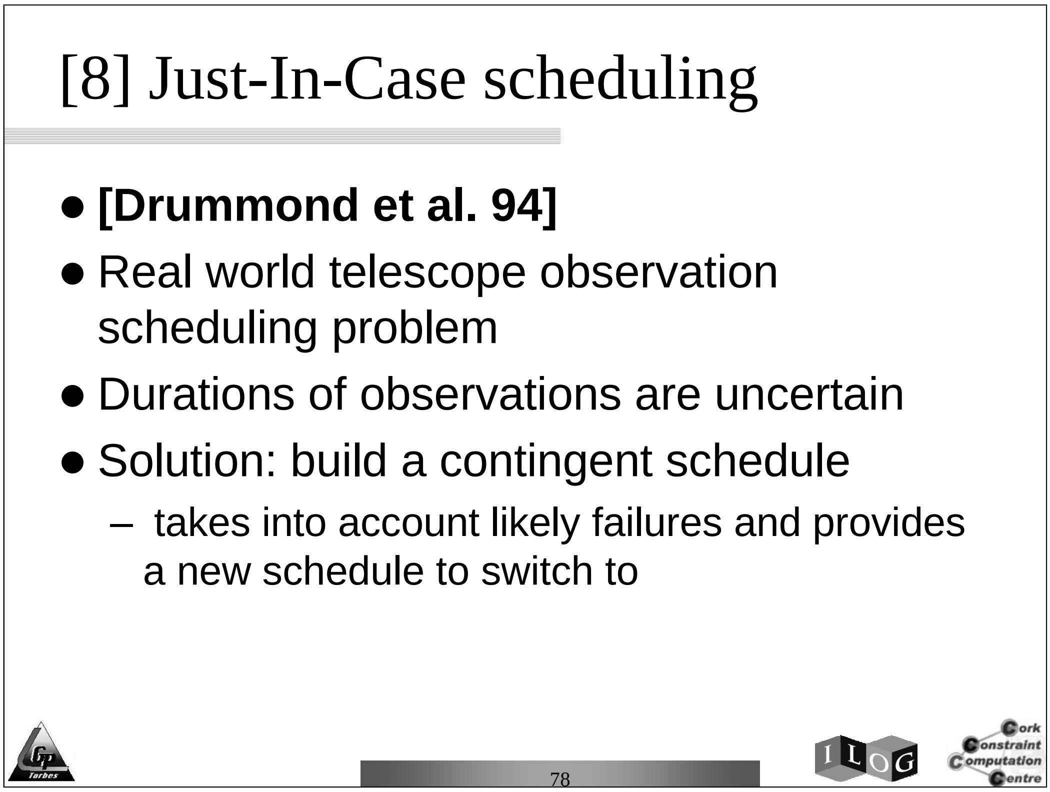
[6] Possibilistic Approach

- **[Dubois et al. 93]**
- Similar to β-robustness with fuzzy durations
- Classical search, but
  - – replace constraint satisfaction requirement by "reasonably sure that no constraint will be violated”
- Realistic approach between
  - – accepting only schedules that are sure to work
  - – accepting a schedule without accounting for possible deviations



[7] Redundancy-based Scheduling

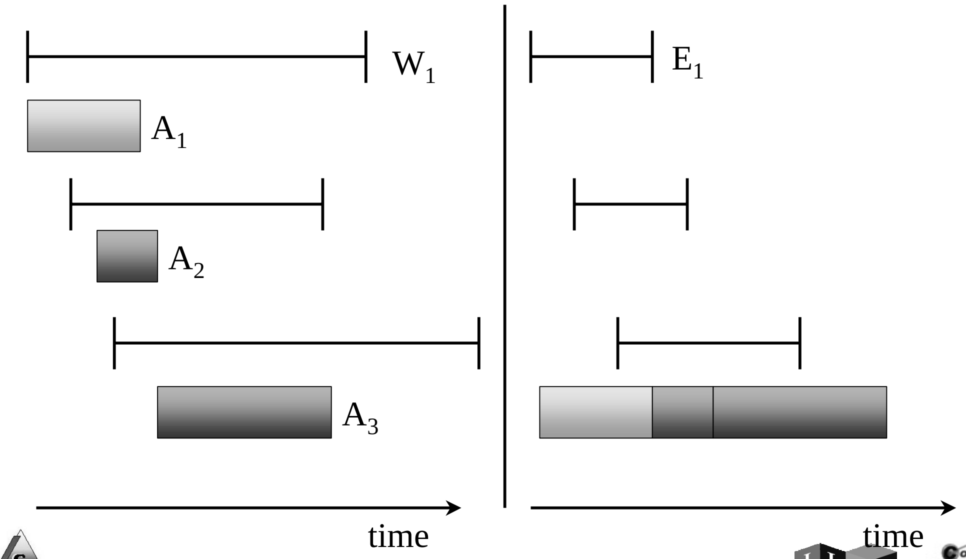
- **[Davenport et al. 01]**
- Add slack to activities on breakable resources
- Simple and pragmatic
- Obvious application to a real problem
- No real theoretical foundation



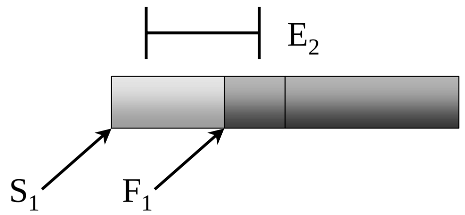
[8] Just-In-Case scheduling

- **[Drummond et al. 94]**
- Real world telescope observation scheduling problem
- Durations of observations are uncertain
- Solution: build a contingent schedule
  - takes into account likely failures and provides a new schedule to switch to

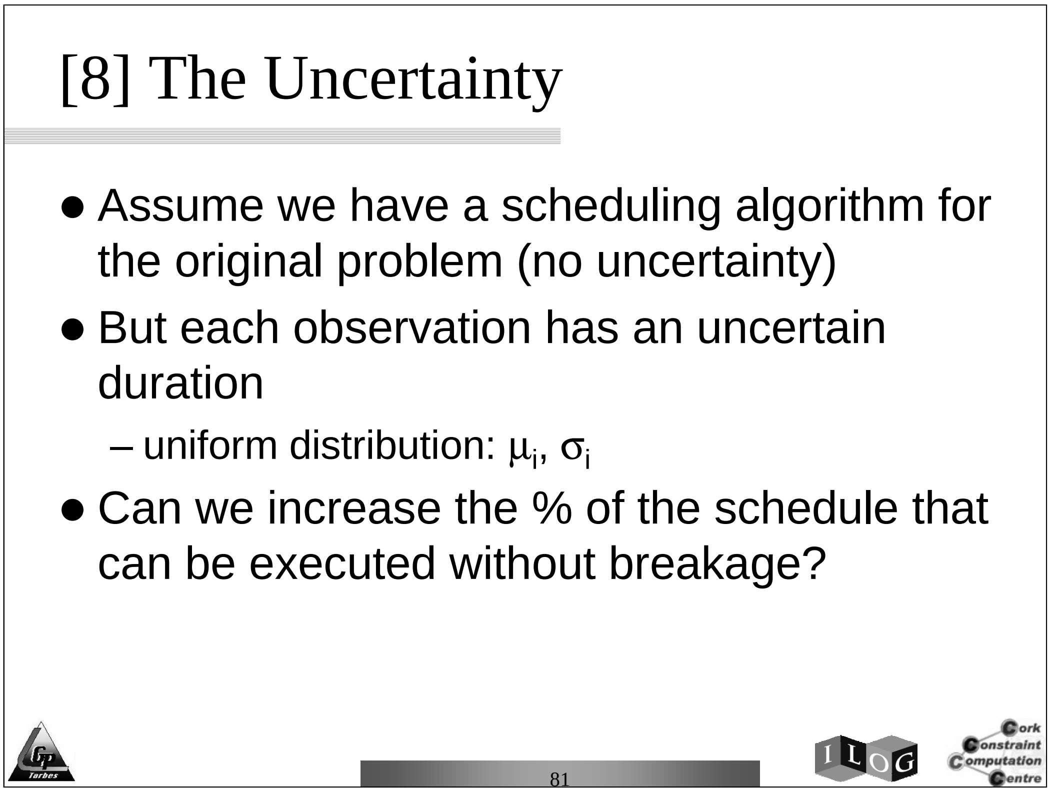
[8] Original Problem



[8] Execution Model



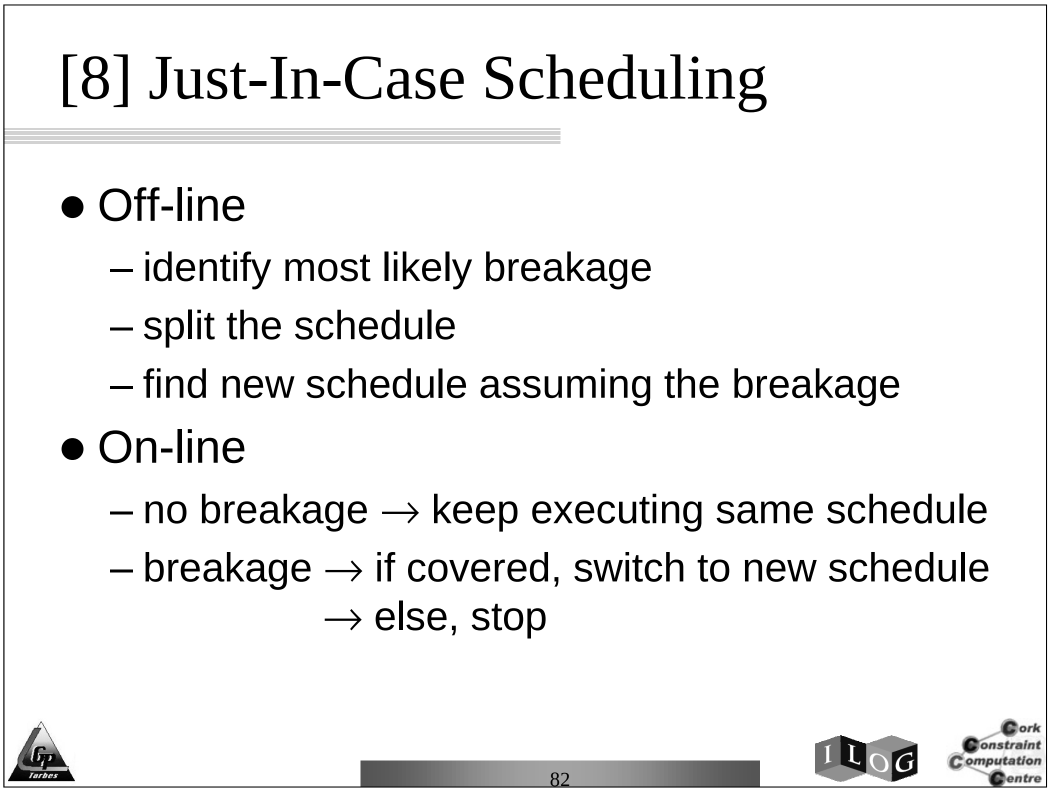
- If $F_1 \in E_2 \rightarrow$ OK
  - – A_2 is executed
- Else → Schedule Breakage
  - – need to reschedule



[8] The Uncertainty

- Assume we have a scheduling algorithm for the original problem (no uncertainty)
- But each observation has an uncertain duration
  - uniform distribution: μ_i, σ_i
- Can we increase the % of the schedule that can be executed without breakage?

Original Problem: find a sequence of observations and assign an enablement interval, E_i, to each observation.

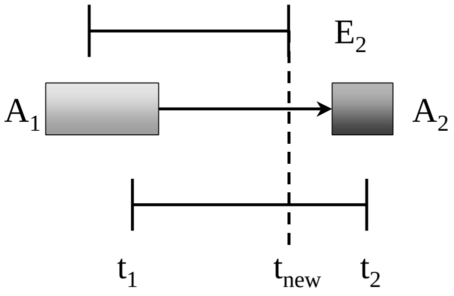


[8] Just-In-Case Scheduling

- Off-line
  - – identify most likely breakage
  - – split the schedule
  - – find new schedule assuming the breakage
- On-line
  - – no breakage → keep executing same schedule
  - – breakage → if covered, switch to new schedule
    → else, stop

In the real application, when a breakage that was not anticipated happens, the dynamic rescheduling is done. For the experiments the focus is the amount of the schedule that can be executed without breakage.

[8] Splitting the Schedule



$[t_1\ t_2)$ are the possible finish times for A_1. In practice: $[S_1 + \mu_1 - \sigma_1, S_1 + \mu_1 + \sigma_1)$

A_2 is the most likely break

If $F_1 \in [t_1, t_{new})$

OK

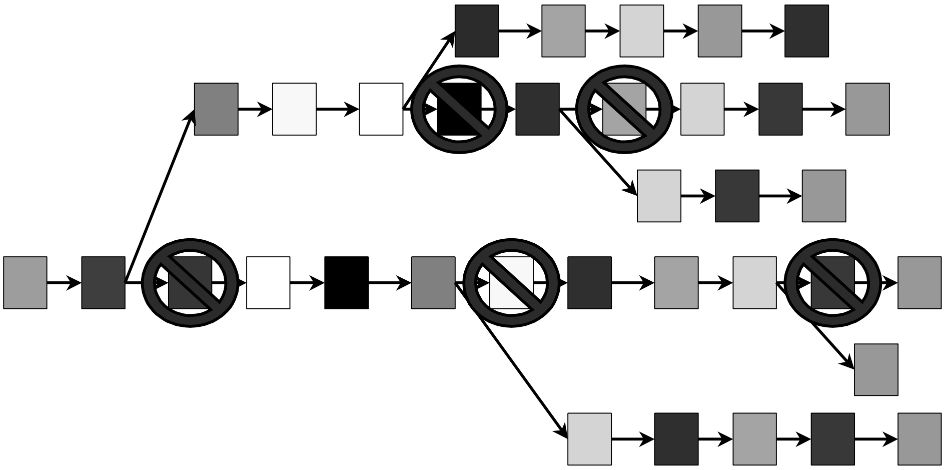
Else

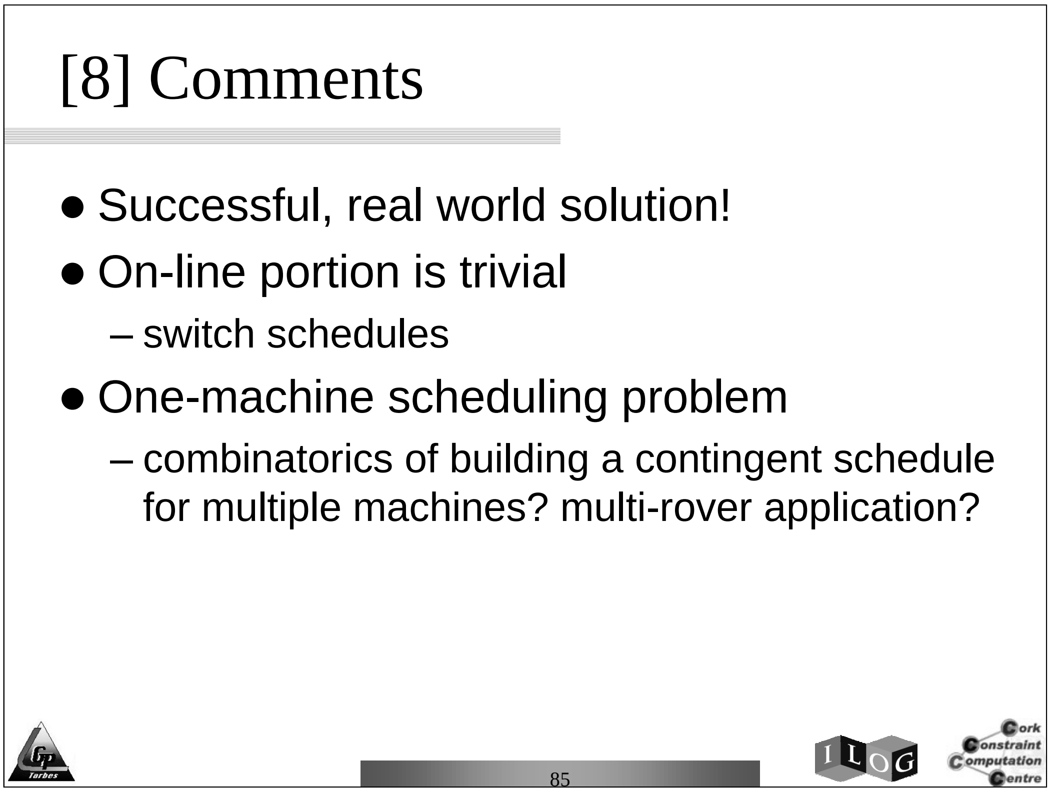
create a new schedule to deal with the case:

$F_1 \in [t_{new}\ t_2)$

(use original scheduler)

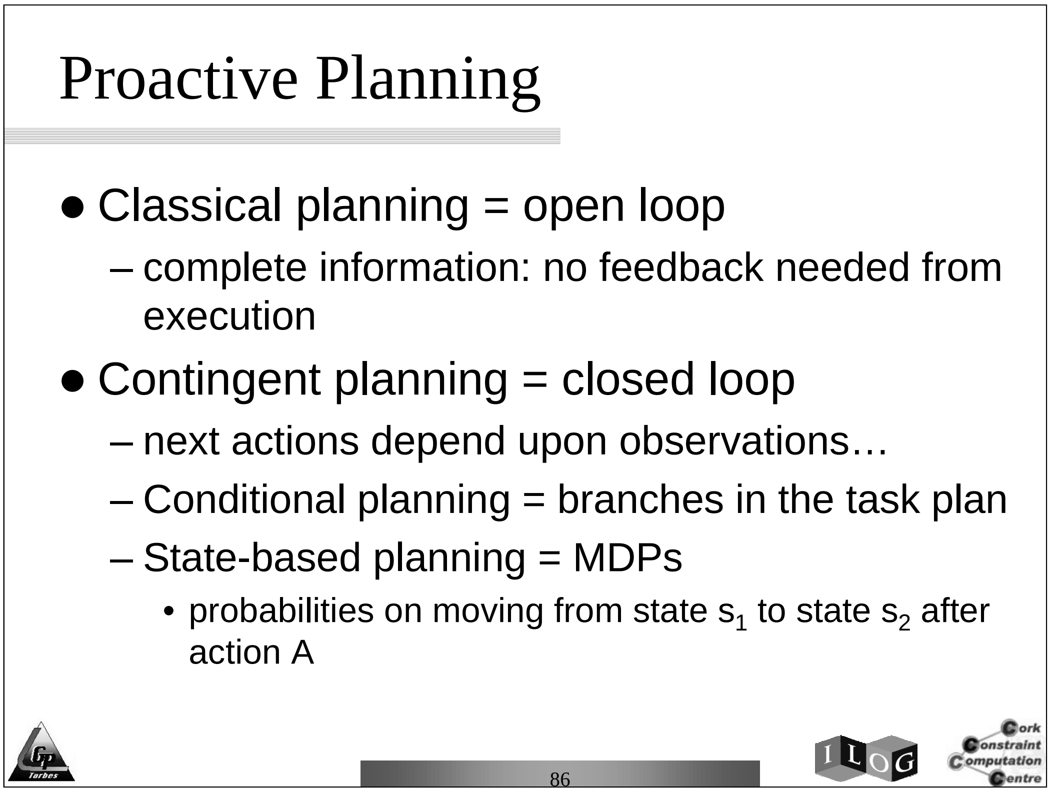
[8] A Multiply Contingent Schedule





[8] Comments

- Successful, real world solution!
- On-line portion is trivial
  – switch schedules
- One-machine scheduling problem
  – combinatorics of building a contingent schedule for multiple machines? multi-rover application?



Proactive Planning

- Classical planning = open loop
  - complete information: no feedback needed from execution
- Contingent planning = closed loop
  - next actions depend upon observations…
  - Conditional planning = branches in the task plan
  - State-based planning = MDPs
    - probabilities on moving from state s_1 to state s_2 after action A

86

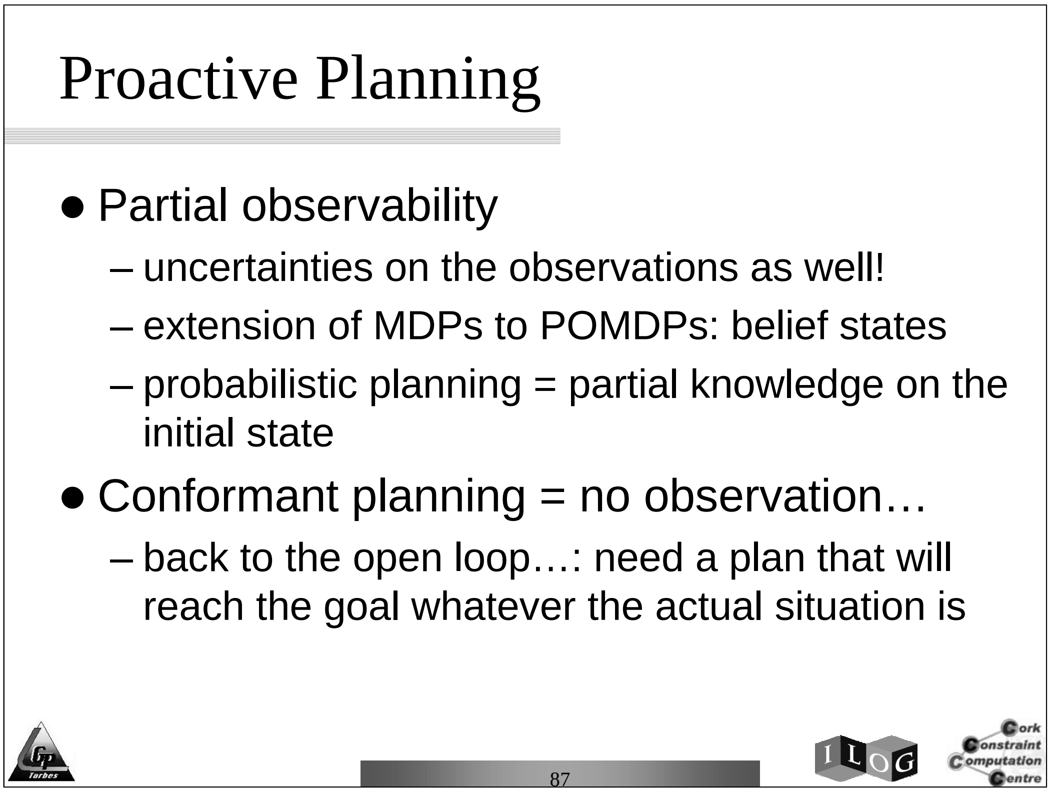
- Closed loop = actions that are outputs of the execution system have consequences that become inputs of the same process.

- Open loop = inputs and outputs are independent.

MDPs (Markov Decision Processes) are related to *decision-theoretic* planning. They are relevant approaches with respect to uncertainty handling in P&S but they will not be addressed in this tutorial. They are very general and are especially well suited to *process-oriented* planning. For detailed analysis of such techniques, please refer to

> C. Boutilier, T. Dean, and S. Hanks. *Decision-theoretic planning: Structural assumptions and computational leverage*. Journal of Artificial Intelligence Research, 11:1--94, 1999.

Briefly speaking, they lie in the general set of approaches in which alternatives are explicitly represented and must be matched at execution time, hence they are very efficient but have the drawback of needing a lot of memory. More compact factored representations exist (intensional state representation, translation into bayesian temporal networks) but are effective only under specific assumptions. MDPs also incorporates probabilities and can hence be used with a combination of *maximal coverage* and *conditional* approaches.



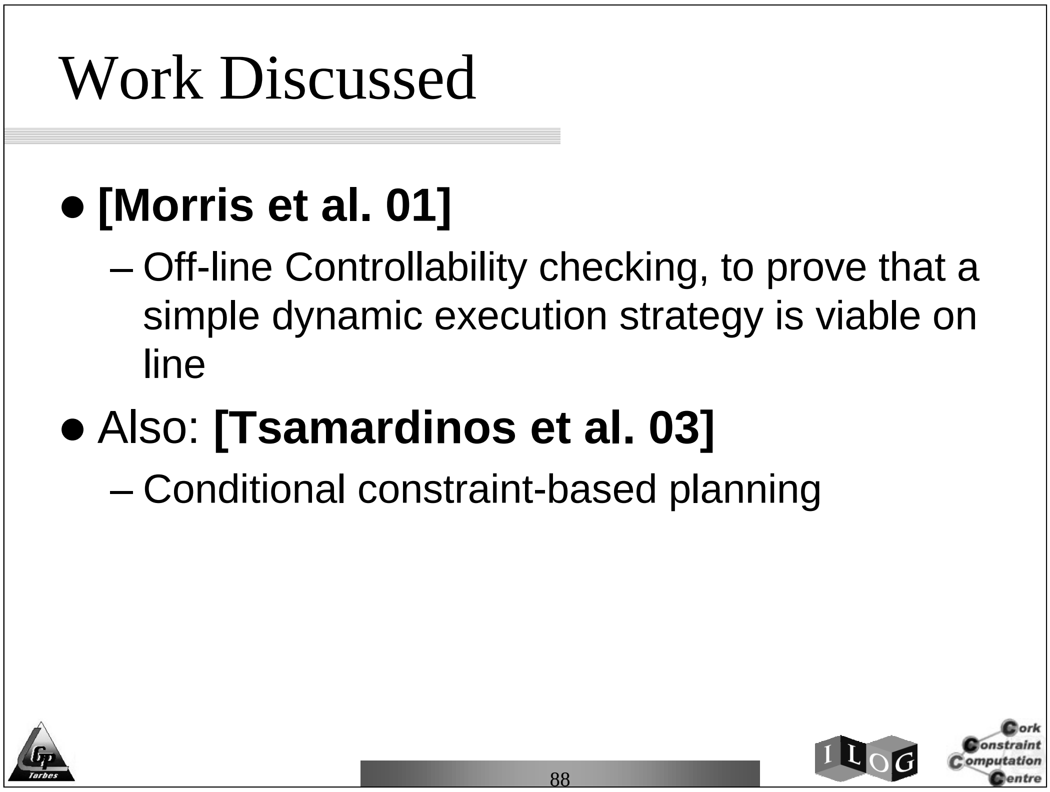
Proactive Planning

- Partial observability
  - uncertainties on the observations as well!
  - extension of MDPs to POMDPs: belief states
  - probabilistic planning = partial knowledge on the initial state
- Conformant planning = no observation…
  - back to the open loop…: need a plan that will reach the goal whatever the actual situation is

Partial observability, probabilistic planning and conformant planning are only cited for situating the other approaches with respect to them. Just recall that in this tutorial we chose to restrict ourselves to full observability.

For an example of conformant planning, see:

D. Smith and D. Weld, *Conformant Graphplan*. Proceedings of the Sixteenth National Conference on Artificial Intelligence, Madison, WI, 1998



Work Discussed

- **[Morris et al. 01]**
  - – Off-line Controllability checking, to prove that a simple dynamic execution strategy is viable on line
- Also: **[Tsamardinos et al. 03]**
  - – Conditional constraint-based planning

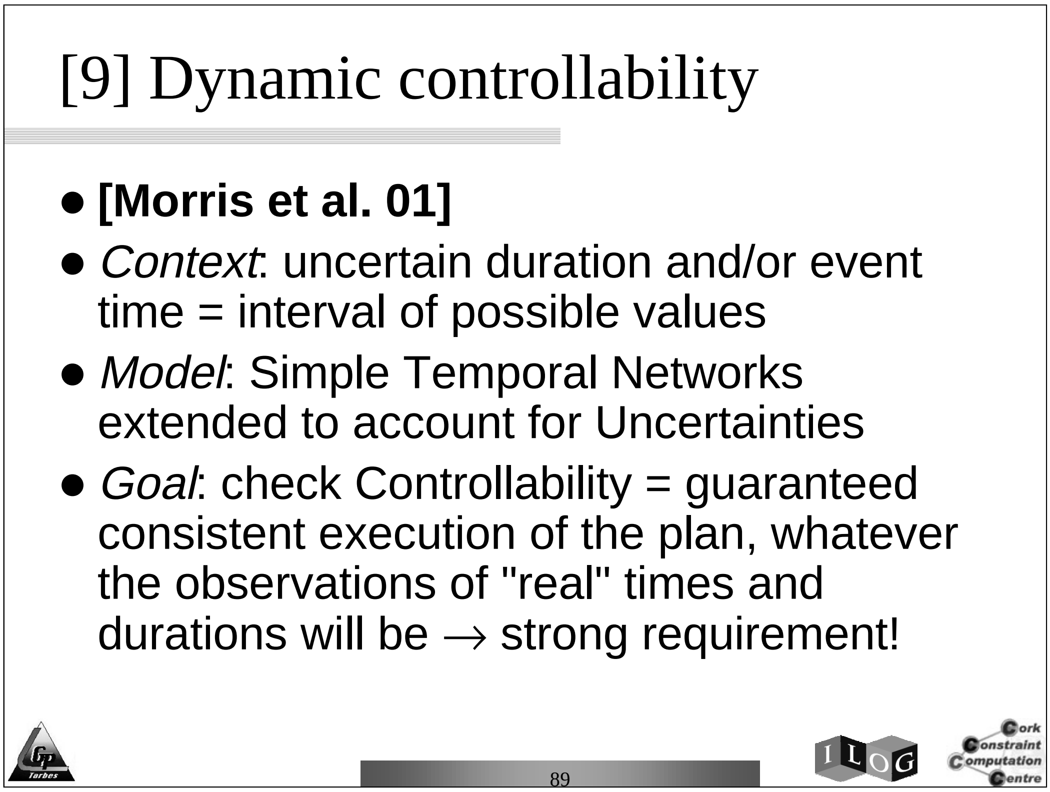
88

9. **[Morris et al. 01]** Morris, P., Muscettola, N. and Vidal, T. *Dynamic Control Of Plans With Temporal Uncertainty*. In Proceedings of the 17th International Joint Conference on A.I. (IJCAI-01). Seattle, 2001.

See also :

[Huguet et al. 02] Huguet, M.J, Lopez, P. and Vidal, T. *Dynamic task sequencing in temporal problems with uncertainty.* AIPS'02 Workshop on On-line Planning and Scheduling, Toulouse, 2002. `http://www.laas.fr/aips/ws-we3.pdf`

10. **[Tsamardinos et al. 03]** Tsamardinos, I., Vidal, T. and Pollack, M.E. *CTP: A New Constraint-Based Formalism for Conditional, Temporal Planning*. CONSTRAINTS, An International Journal, vol. 8:4, 2003.

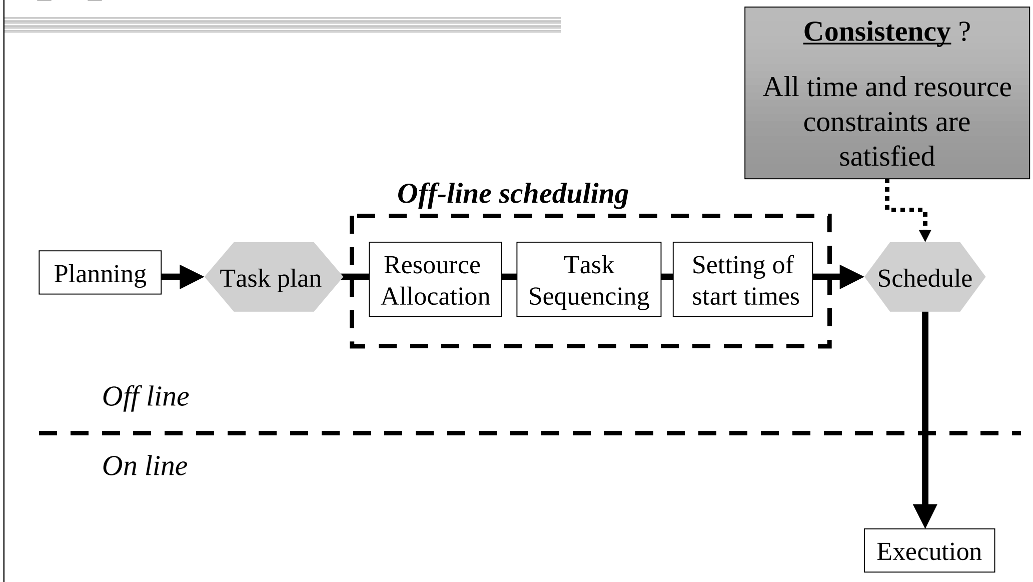


[9] Dynamic controllability

- **[Morris et al. 01]**
- *Context*: uncertain duration and/or event time = interval of possible values
- *Model*: Simple Temporal Networks extended to account for Uncertainties
- *Goal*: check Controllability = guaranteed consistent execution of the plan, whatever the observations of "real" times and durations will be → strong requirement!

89

[9] Classical constraint-based P&S



Consistency ?

All time and resource constraints are satisfied

Off-line scheduling

Planning

Task plan

Resource Allocation

Task Sequencing

Setting of start times

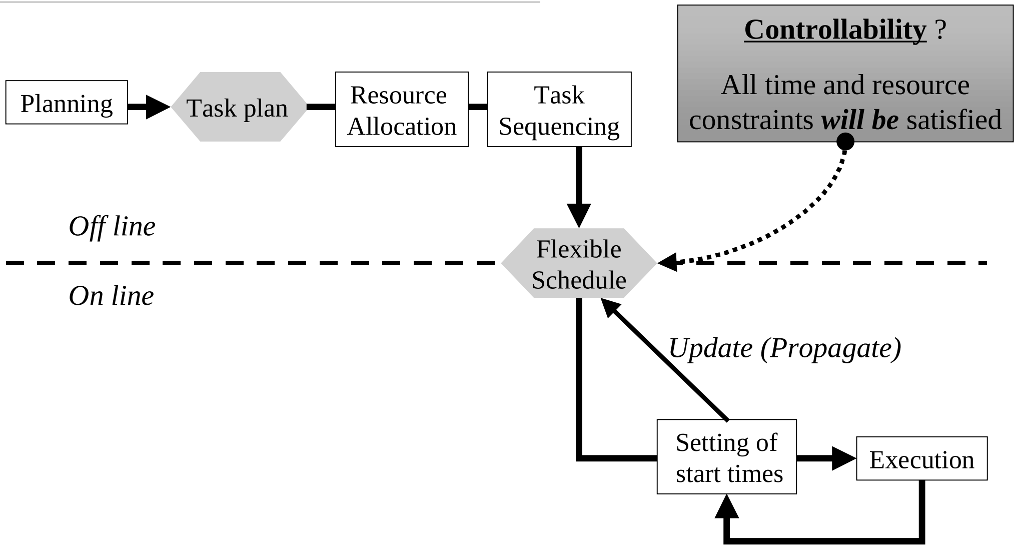
Schedule

Off line

On line

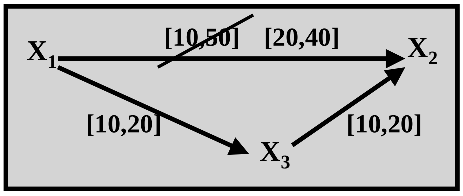
Execution

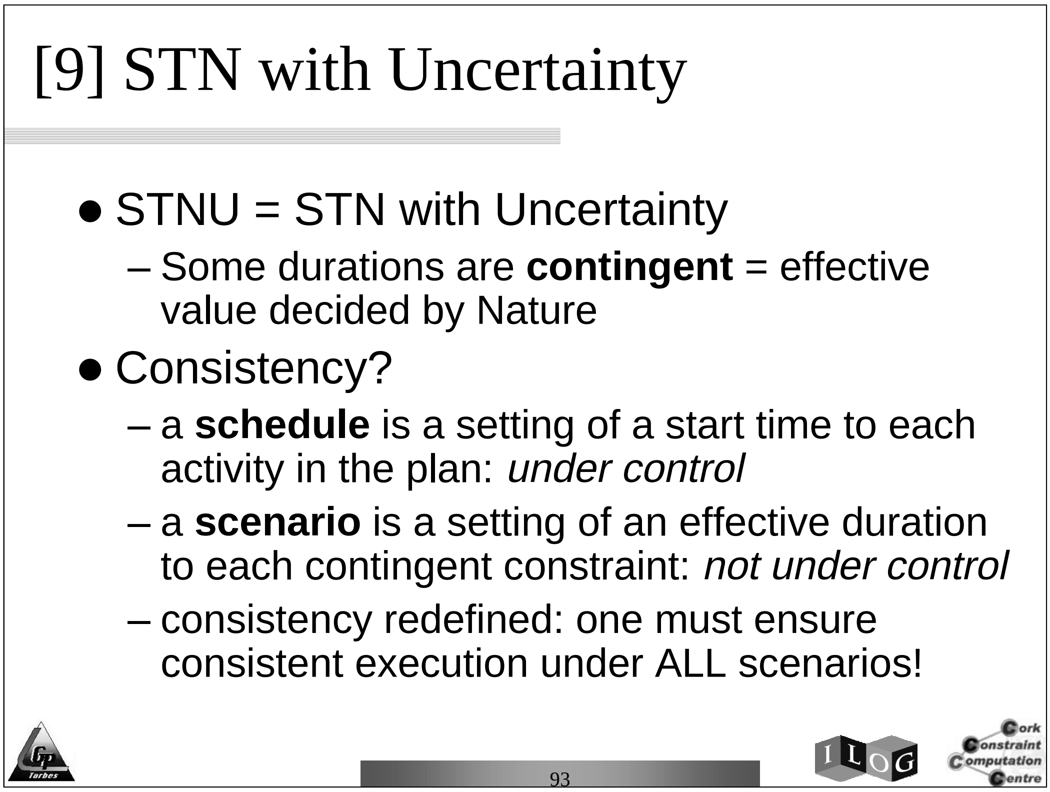
[9] Dynamic scheduling



[9] Temporal Constraint Problems

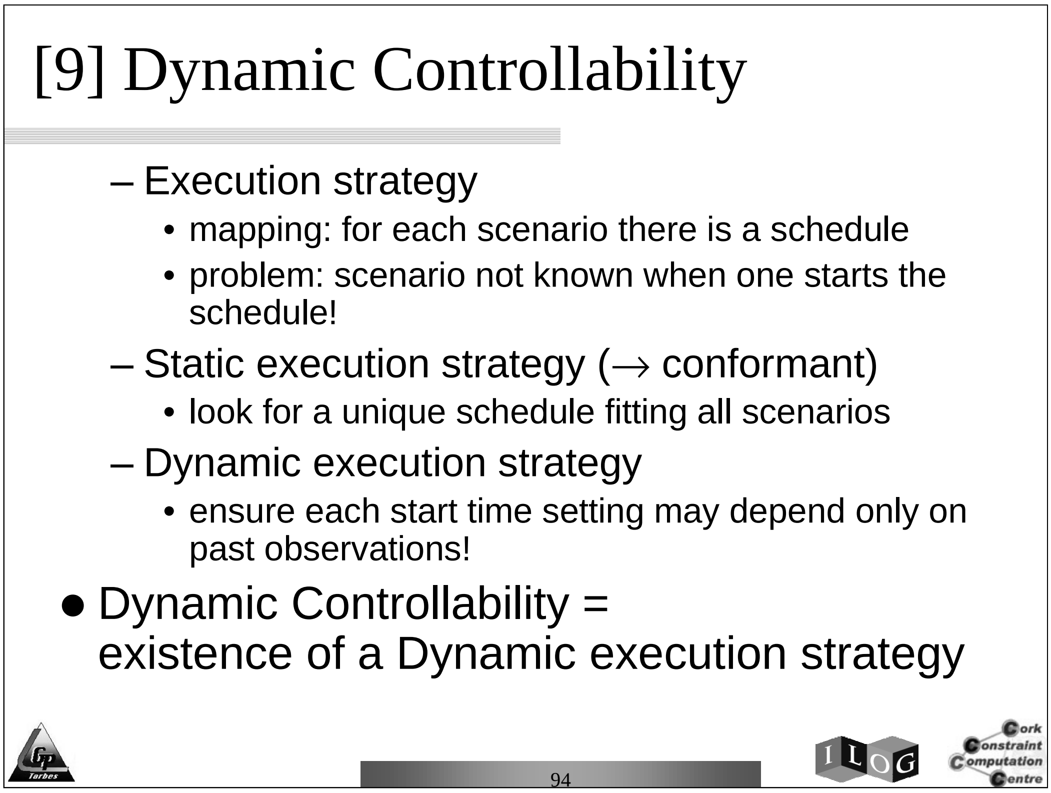
- Simple Temporal Network (STN)
  - – Nodes = Time-points; Edges = Durations
  - – Constraint propagation in $O(n^3)$
    - → filters out durations that would lead to inconsistency
    - → ensures consistent execution





[9] STN with Uncertainty

- STNU = STN with Uncertainty
  - Some durations are **contingent** = effective value decided by Nature
- Consistency?
  - a **schedule** is a setting of a start time to each activity in the plan: *under control*
  - a **scenario** is a setting of an effective duration to each contingent constraint: *not under control*
  - consistency redefined: one must ensure consistent execution under ALL scenarios!



[9] Dynamic Controllability

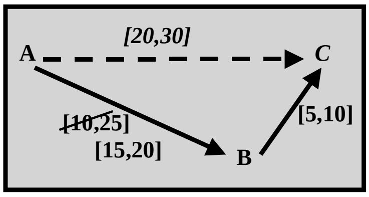
- – Execution strategy
  - mapping: for each scenario there is a schedule
  - problem: scenario not known when one starts the schedule!
- – Static execution strategy (→ conformant)
  - look for a unique schedule fitting all scenarios
- – Dynamic execution strategy
  - ensure each start time setting may depend only on past observations!

- Dynamic Controllability =
existence of a Dynamic execution strategy

[9] DC checking: examples

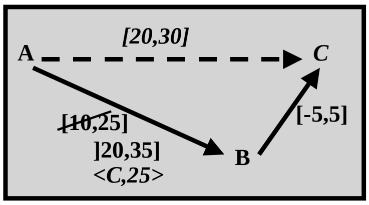
– New propagation algorithm in O(n^3)

- Case where start time B must not depend on AC



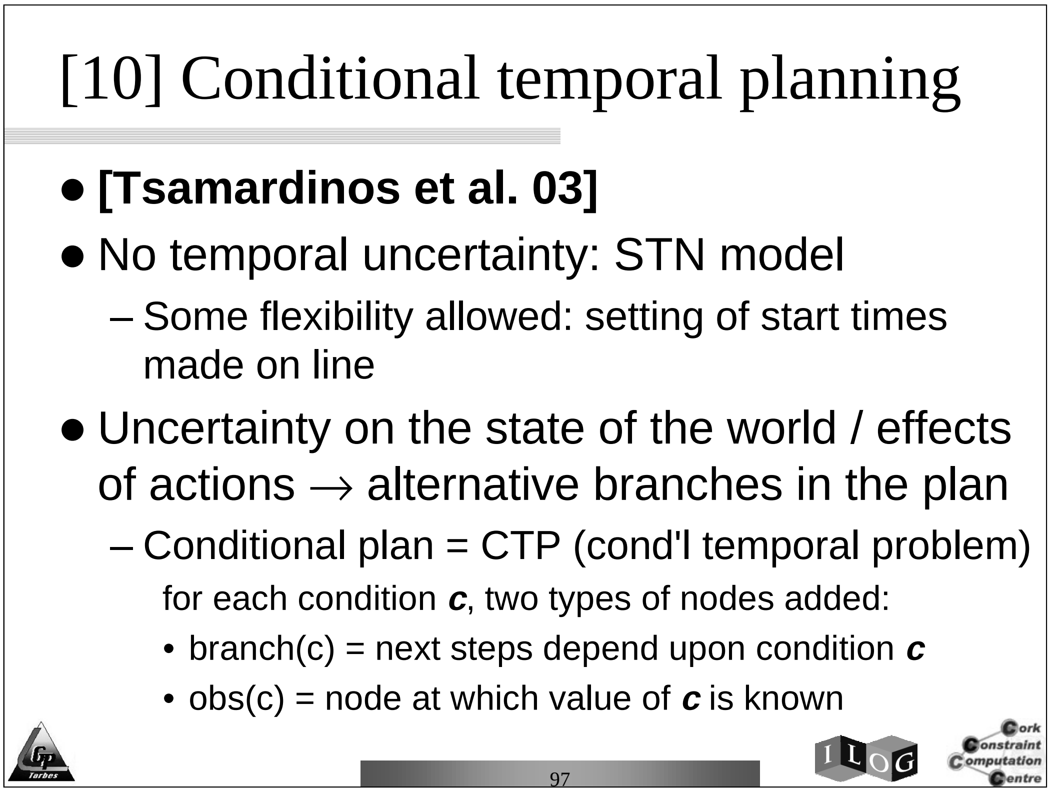
- Case where start time B may or not depend on AC

Wait for C at least 25



[9] Comments

- The requirement is strong: only plans that are proved to safely execute are accepted
  - might be combined with probabilistic or fuzzy modelling to accept "reasonably sure" plans
- The off-line checking process is inspired by how the plan will be dynamically executed
  - provides strong guarantees upon execution
- Simple network propagation
  - low cost both in time and memory!

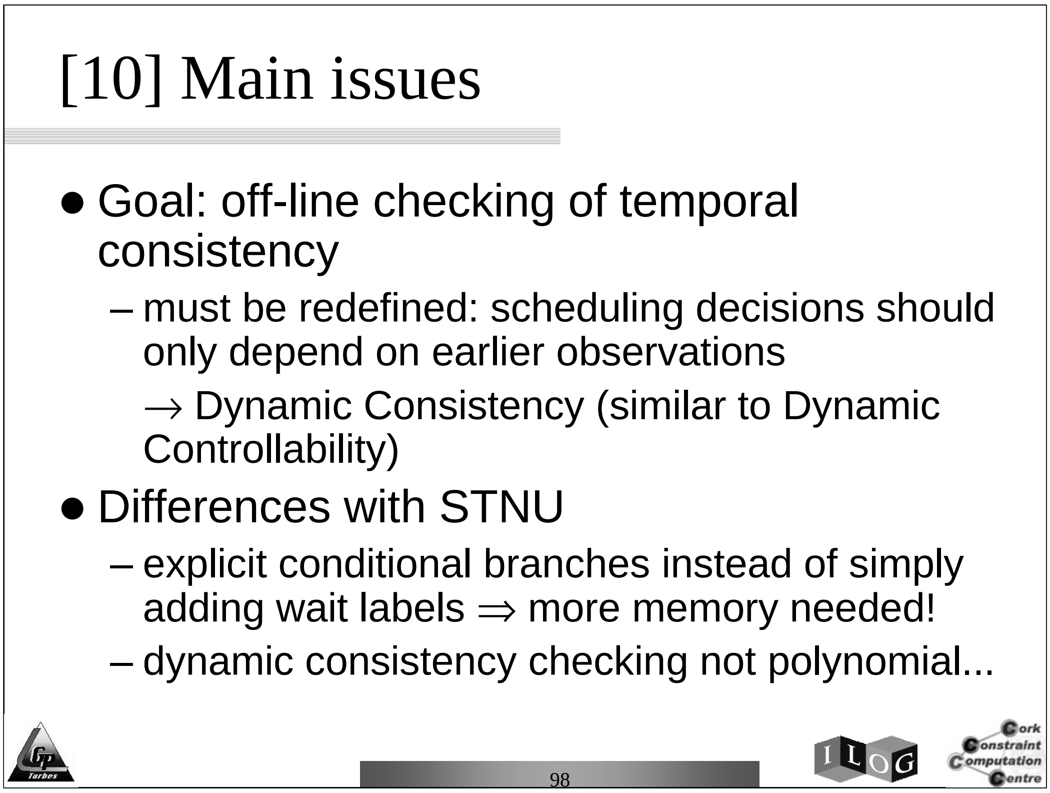


[10] Conditional temporal planning

- **[Tsamardinos et al. 03]**
- No temporal uncertainty: STN model
  - Some flexibility allowed: setting of start times made on line
- Uncertainty on the state of the world / effects of actions → alternative branches in the plan
  - Conditional plan = CTP (cond'l temporal problem)

    for each condition ***c***, two types of nodes added:

    - branch(c) = next steps depend upon condition ***c***
    - obs(c) = node at which value of ***c*** is known



[10] Main issues

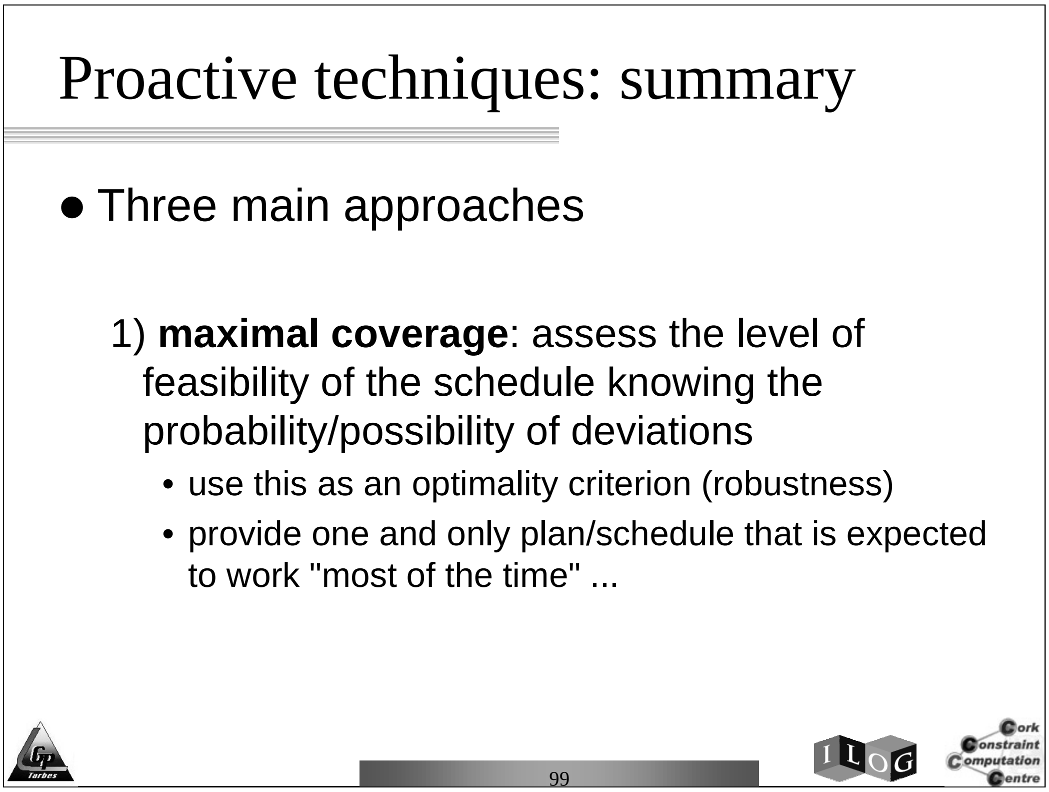
- Goal: off-line checking of temporal consistency
  - – must be redefined: scheduling decisions should only depend on earlier observations

    → Dynamic Consistency (similar to Dynamic Controllability)
- Differences with STNU
  - – explicit conditional branches instead of simply adding wait labels ⇒ more memory needed!
  - – dynamic consistency checking not polynomial...

Dynamic consistency is complex mainly because, when observation nodes and others are unordered, one gets disjunctive constraints… The resulting problem to solve is a DTP (Disjunctive Temporal Problem) solved by dedicated solvers.

A basic reference on DTPs is

K. Stergiou and M. Koubarakis, *Backtracking algorithms for disjunctions of temporal constraints*. Artificial Intelligence 120, 81-117, 2000.

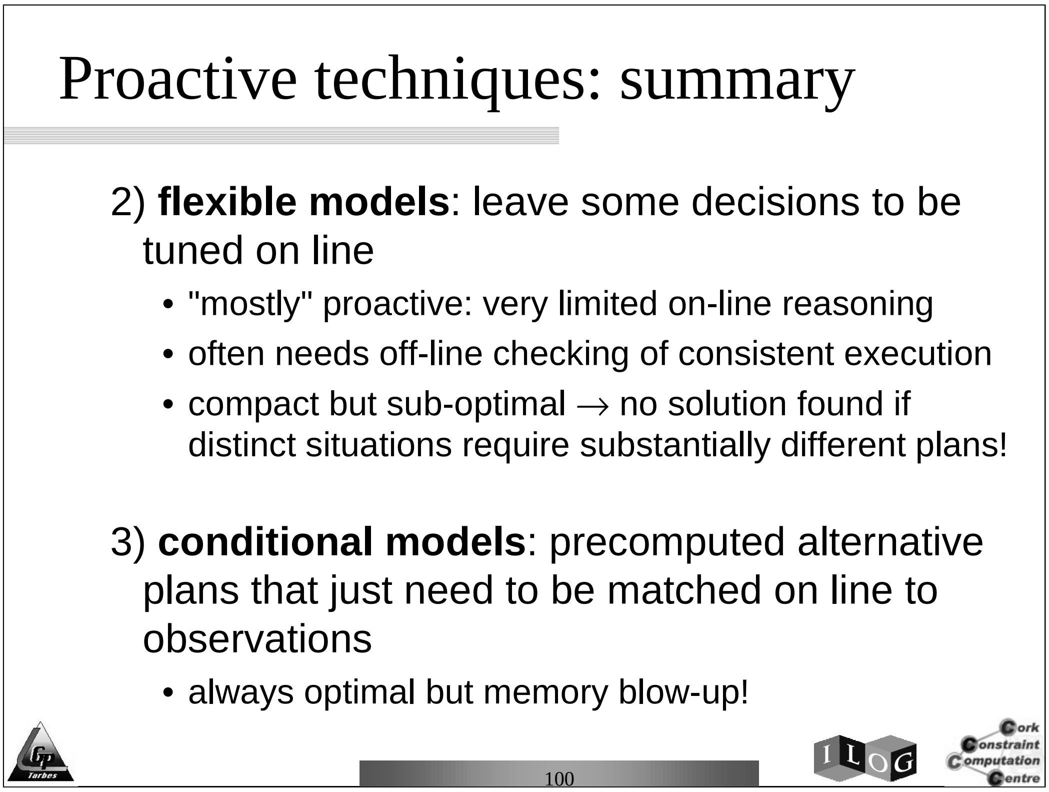


Proactive techniques: summary

- Three main approaches

  1) **maximal coverage**: assess the level of feasibility of the schedule knowing the probability/possibility of deviations
  - use this as an optimality criterion (robustness)
  - provide one and only plan/schedule that is expected to work "most of the time" ...

[Daniels & Carrillo 97] and [Dubois et al. 93] are examples of the first line of approaches.



Proactive techniques: summary

2) **flexible models**: leave some decisions to be tuned on line

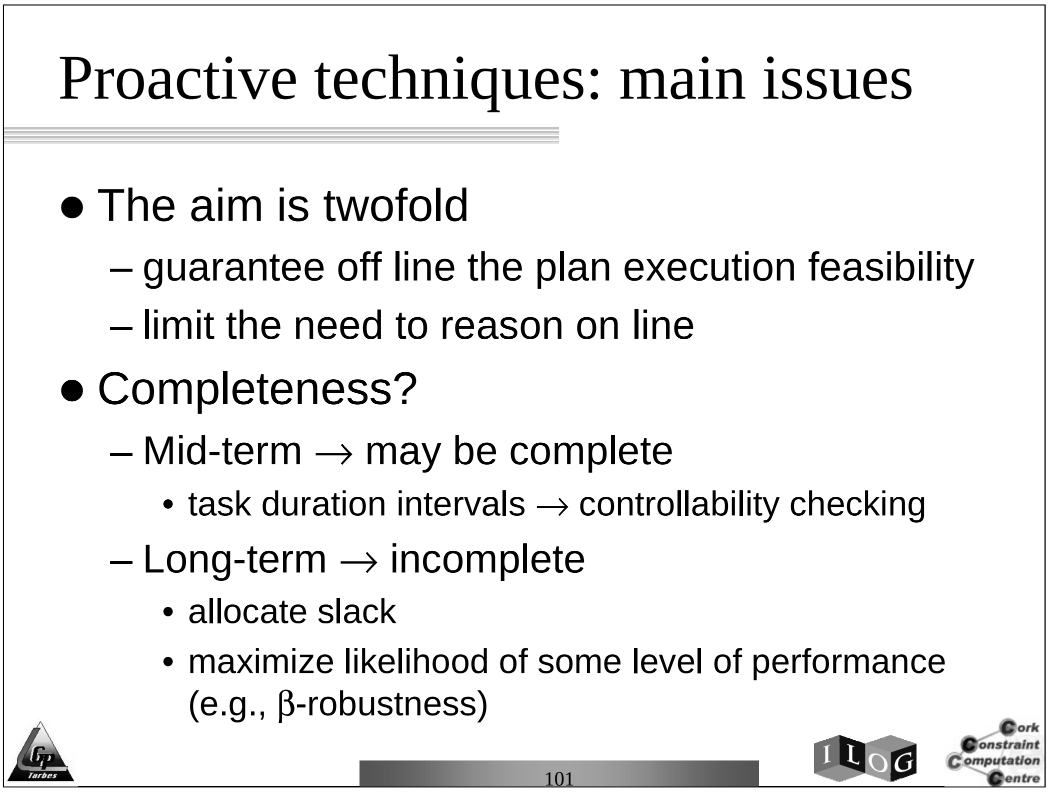
- "mostly" proactive: very limited on-line reasoning
- often needs off-line checking of consistent execution
- compact but sub-optimal → no solution found if distinct situations require substantially different plans!

3) **conditional models**: precomputed alternative plans that just need to be matched on line to observations

- always optimal but memory blow-up!

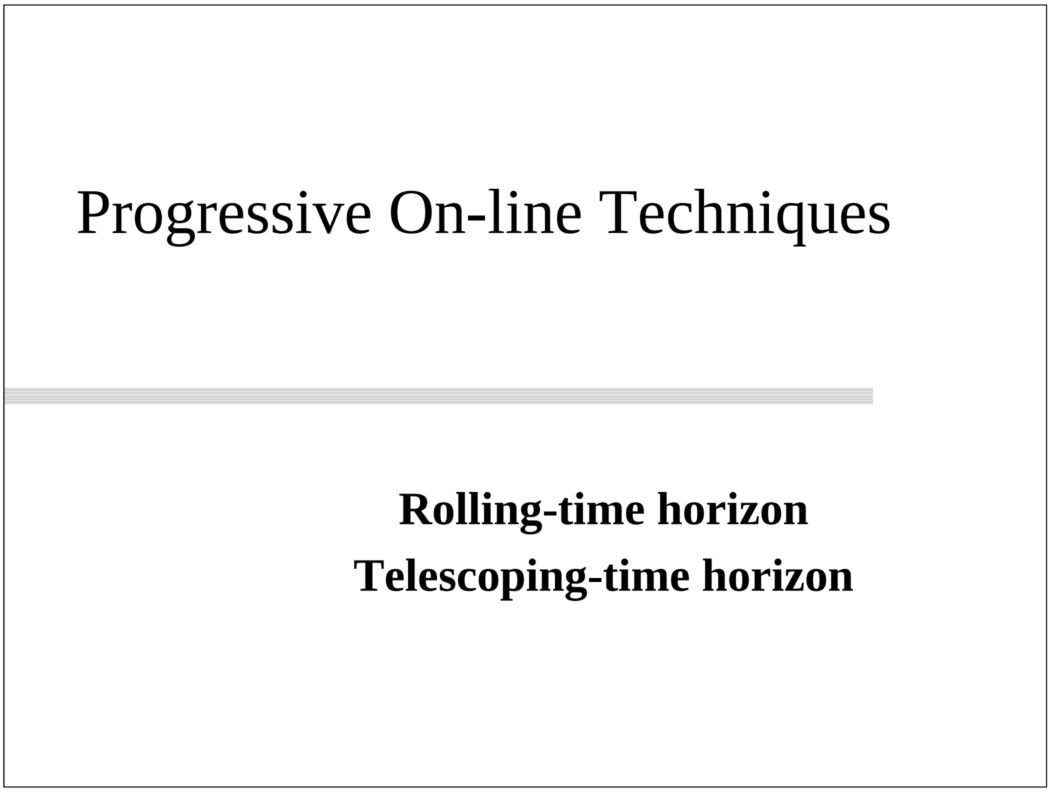
[Davenport et al. 01] and [Morris et al. 01] are examples of the second line of approaches.

[Drummond et al. 94] and [Tsamardinos et al. 03] are examples of the third line of approaches.



Proactive techniques: main issues

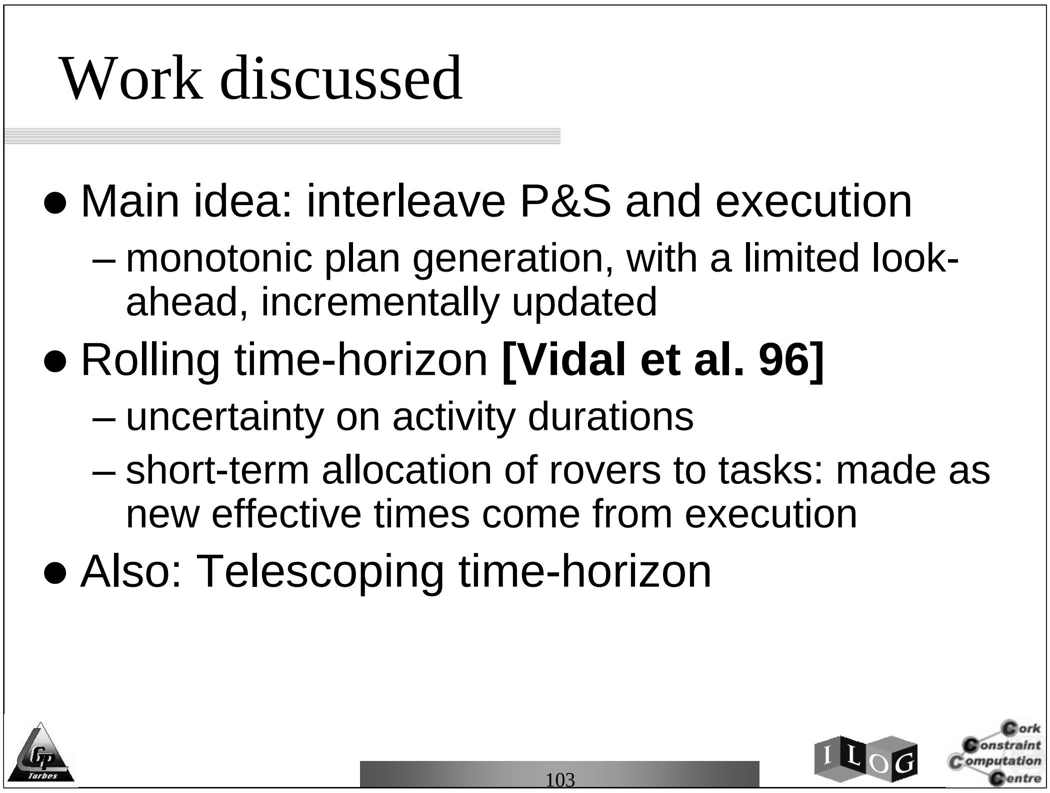
- The aim is twofold
  - guarantee off line the plan execution feasibility
  - limit the need to reason on line
- Completeness?
  - Mid-term → may be complete
    - task duration intervals → controllability checking
  - Long-term → incomplete
    - allocate slack
    - maximize likelihood of some level of performance (e.g., β-robustness)



Progressive On-line Techniques

Rolling-time horizon

Telescoping-time horizon



Work discussed

- Main idea: interleave P&S and execution
  - – monotonic plan generation, with a limited look-ahead, incrementally updated
- Rolling time-horizon **[Vidal et al. 96]**
  - – uncertainty on activity durations
  - – short-term allocation of rovers to tasks: made as new effective times come from execution
- Also: Telescoping time-horizon

103

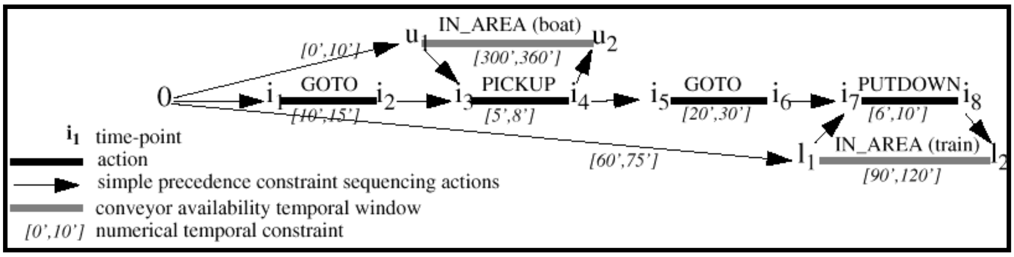
11. **[Vidal et al. 96]** Vidal, T., Ghallab, M. and Alami, R. *Incremental Mission Allocation to a Large Team of Robots*. In Proceedings IEEE Robotics and Automation, Minneapolis, 1996.

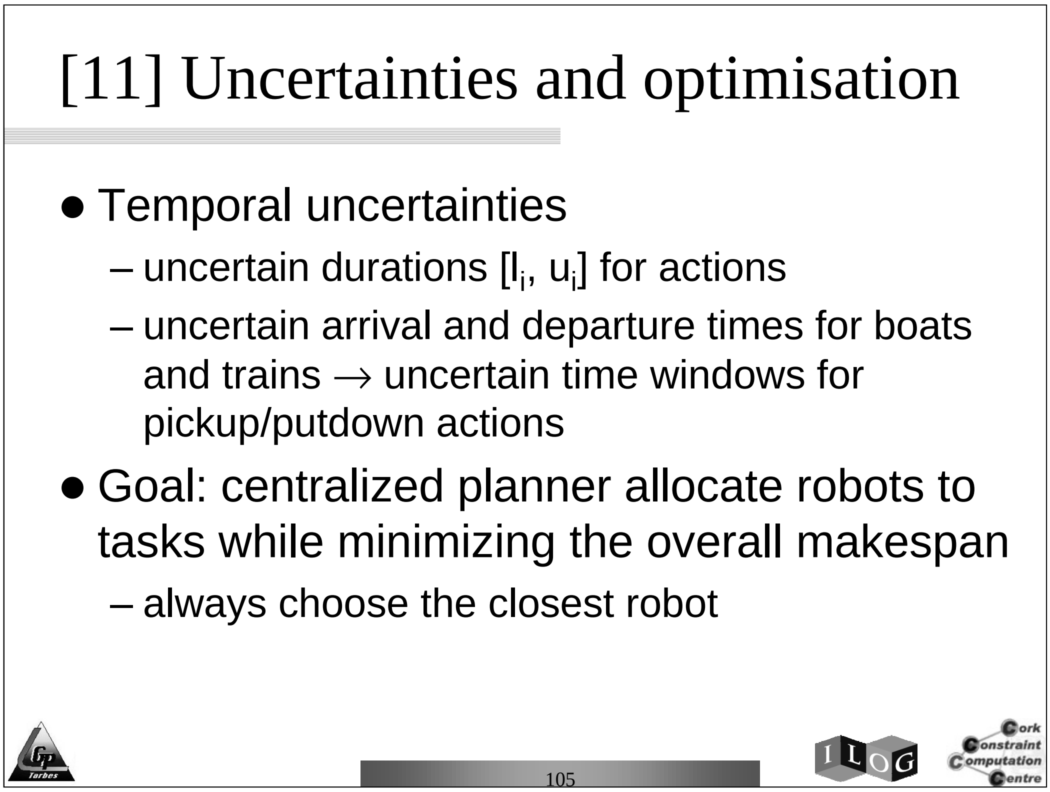
Telescoping time horizon means a global plan is generated but it is only detailed in the short range, maintaining an incomplete or abstract plan in the longer range. The plan is further detailed as far as execution progresses.

One can think of using different temporal granularities (see for instance Thomas Dean, *Using temporal hierarchies to efficiently maintain large temporal databases*. Journal of the ACM 36(4), 1989) or of using a hierarchical representation of activities, as can be found in HTN (Hierarchical Temporal Networks) planners.

[11] Short-term resource allocation

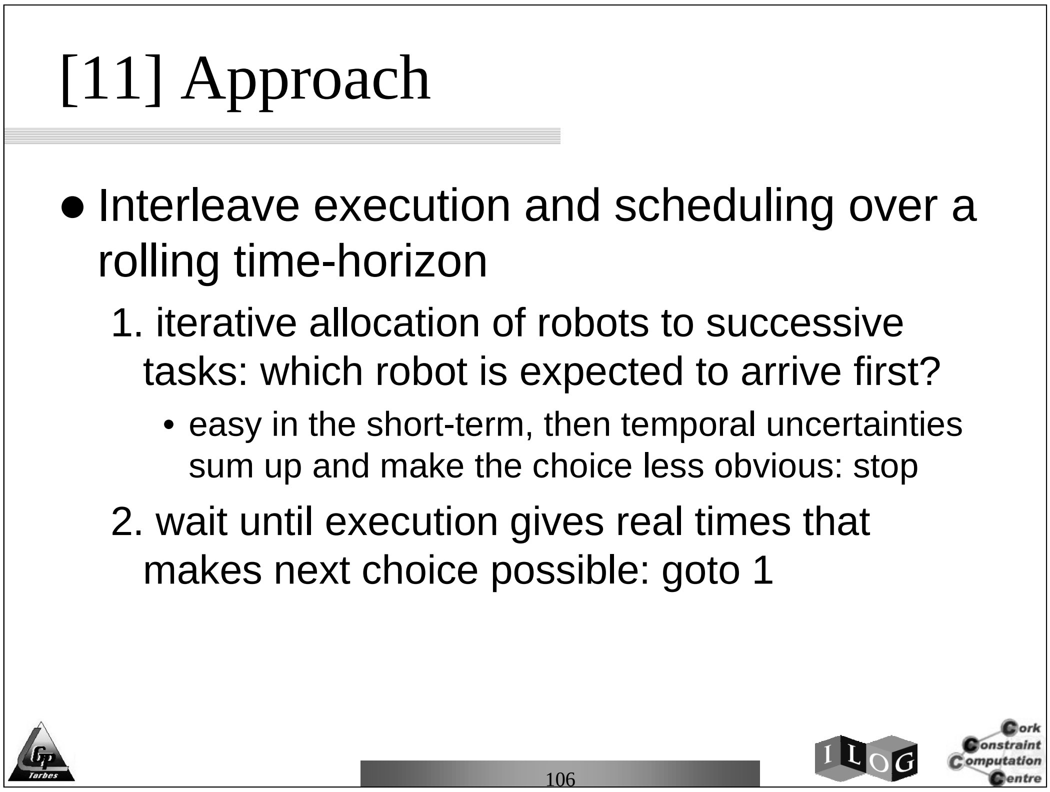
- **[Vidal et al. 96]**: MARTHA Esprit project
- Context: autonomous robots in a harbour environment
  - – repetitive storehouse tasks to move a container from a location to another: four basic actions





[11] Uncertainties and optimisation

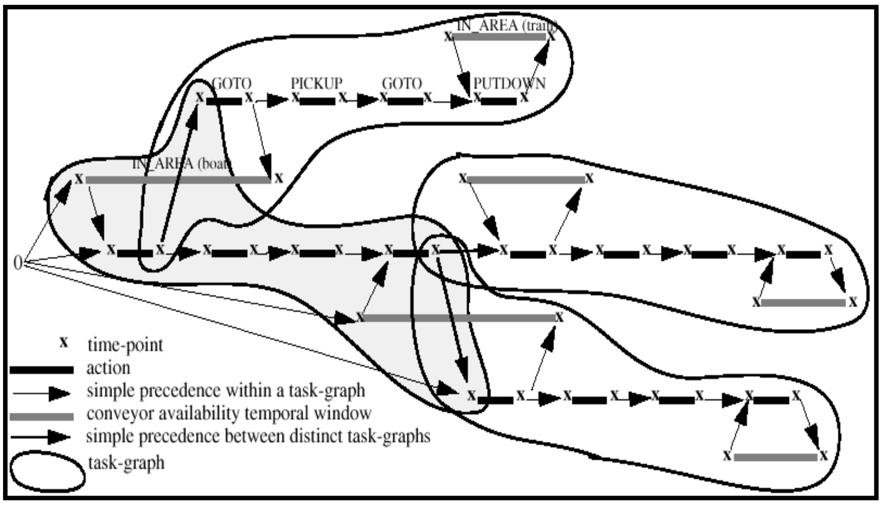
- Temporal uncertainties
  - uncertain durations [l_i, u_i] for actions
  - uncertain arrival and departure times for boats and trains → uncertain time windows for pickup/putdown actions
- Goal: centralized planner allocate robots to tasks while minimizing the overall makespan
  - always choose the closest robot

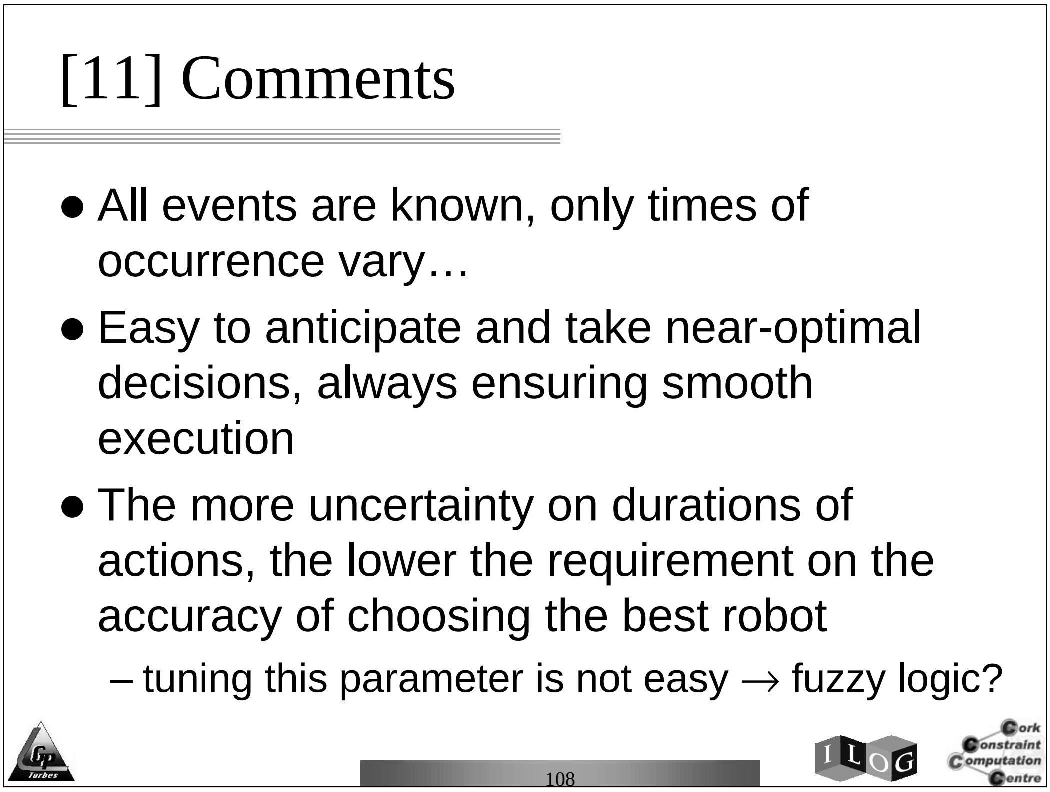


[11] Approach

- Interleave execution and scheduling over a rolling time-horizon
  1. iterative allocation of robots to successive tasks: which robot is expected to arrive first?
     - easy in the short-term, then temporal uncertainties sum up and make the choice less obvious: stop
  2. wait until execution gives real times that makes next choice possible: goto 1

[11] Ordering task-graphs

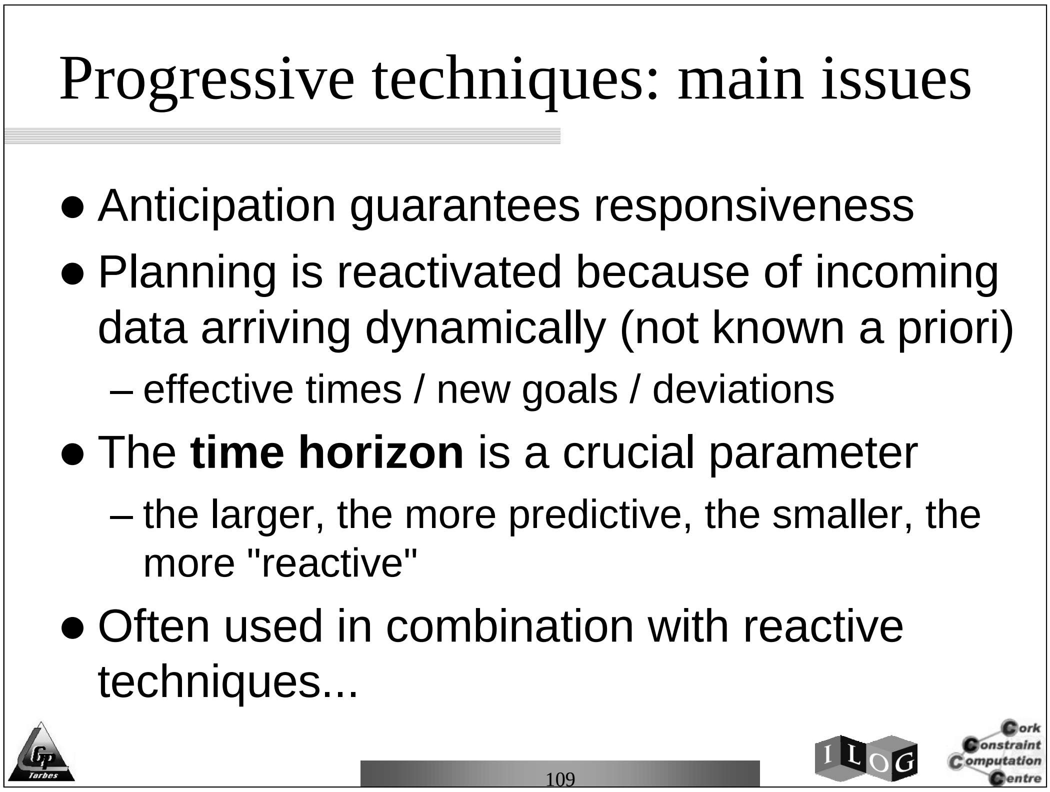

IN_AREA (train)
GOTO
PICKUP
GOTO
PUTDOWN
IN_AREA (boat)
0
x
time-point
action
simple precedence within a task-graph
conveyor availability temporal window
simple precedence between distinct task-graphs
task-graph



[11] Comments

- All events are known, only times of occurrence vary…
- Easy to anticipate and take near-optimal decisions, always ensuring smooth execution
- The more uncertainty on durations of actions, the lower the requirement on the accuracy of choosing the best robot
  - tuning this parameter is not easy → fuzzy logic?

The accuracy requirement was straightforward: just measure the overlapping of the intervals of possible arrival times of the two "best" robots, if this overlapping exceeds a given threshold then the choice is not accurate enough. Moreover, in order to avoid dead ends, the choice was always enforced if the best robot to allocate was idle.



Progressive techniques: main issues

- Anticipation guarantees responsiveness
- Planning is reactivated because of incoming data arriving dynamically (not known a priori)
  - – effective times / new goals / deviations
- The **time horizon** is a crucial parameter
  - – the larger, the more predictive, the smaller, the more "reactive"
- Often used in combination with reactive techniques...

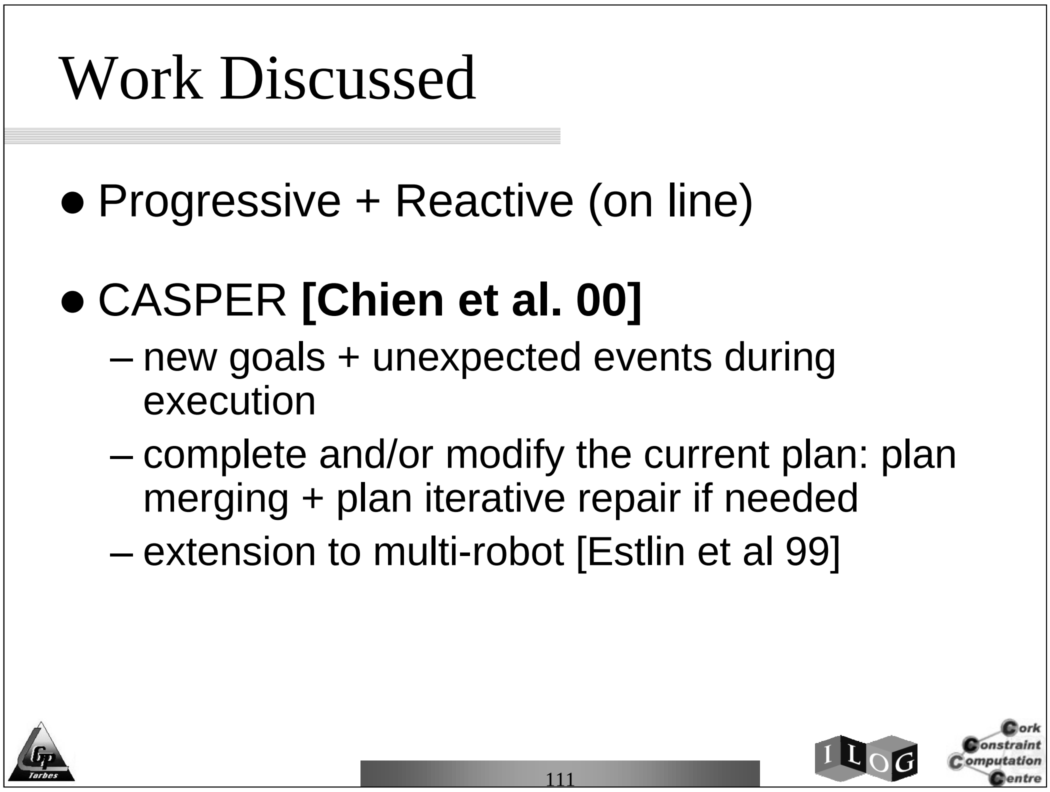
109

A very small time horizon accounts to a *purely reactive* behavior with *no indicative schedule*. Therefore reaction here is not destructive since there is no schedule to revise, but it consists in simply scheduling the next step. As said before, these techniques are known in production scheduling as *dispatching techniques*.

Mixed Approaches

Continuous + Reactive

Proactive + Reactive



Work Discussed

- Progressive + Reactive (on line)
- CASPER **[Chien et al. 00]**
  - new goals + unexpected events during execution
  - complete and/or modify the current plan: plan merging + plan iterative repair if needed
  - extension to multi-robot [Estlin et al 99]

111

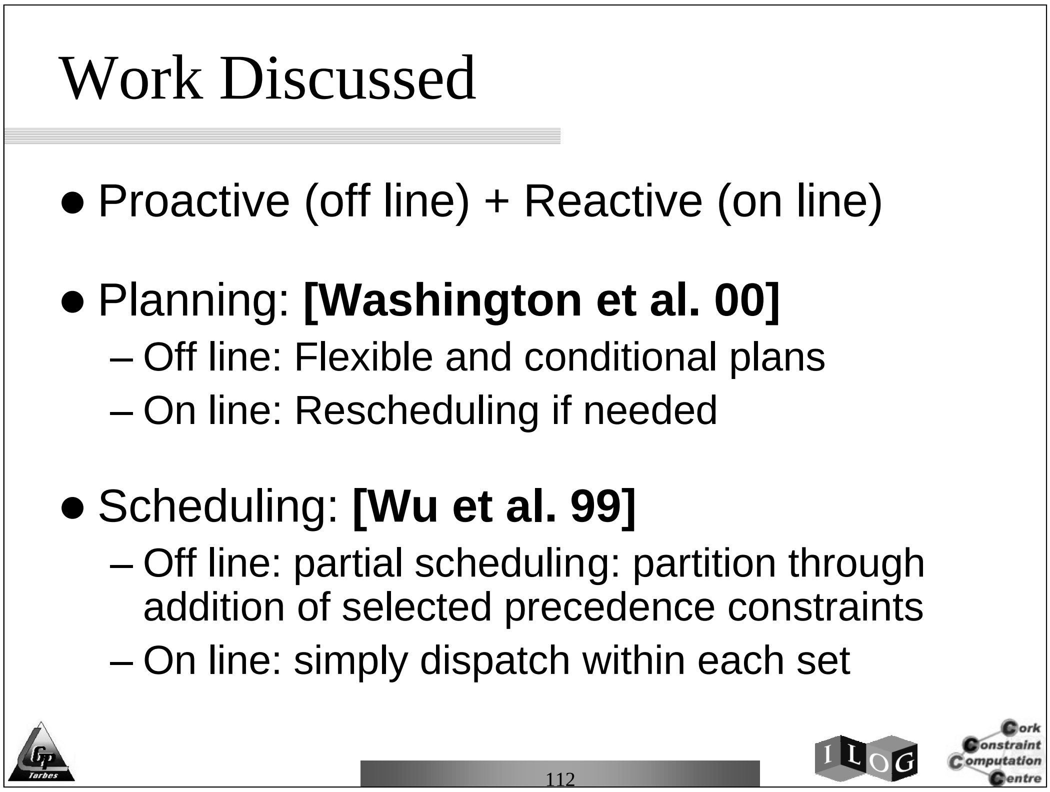
12. **[Chien et al. 00]** Chien, S., Knight, R., Stechert, A., Sherwood, R. and Rabideau, R. *Using Iterative Repair to Improve the Responsiveness of Planning and Scheduling*. In Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling (AIPS'2000), Breckenridge, 2000.

http://www-aig.jpl.nasa.gov/public/home/chien/home.html

See also:

[Estlin et al 99] Estlin, T., Rabideau, G., Mutz, D. and Chien, S. *Using Continuous Planning Techniques to Coordinate Multiple Rovers*. IJCAI-99 Workshop on Scheduling and Planning meet Real-time Monitoring in a Dynamic and Uncertain World, Stockholm, 1999. H

http://www.enit.fr/~Thierry/WsIjcai99/Papiers/estlin.ps



Work Discussed

- Proactive (off line) + Reactive (on line)
- Planning: **[Washington et al. 00]**
  - Off line: Flexible and conditional plans
  - On line: Rescheduling if needed
- Scheduling: **[Wu et al. 99]**
  - Off line: partial scheduling: partition through addition of selected precedence constraints
  - On line: simply dispatch within each set

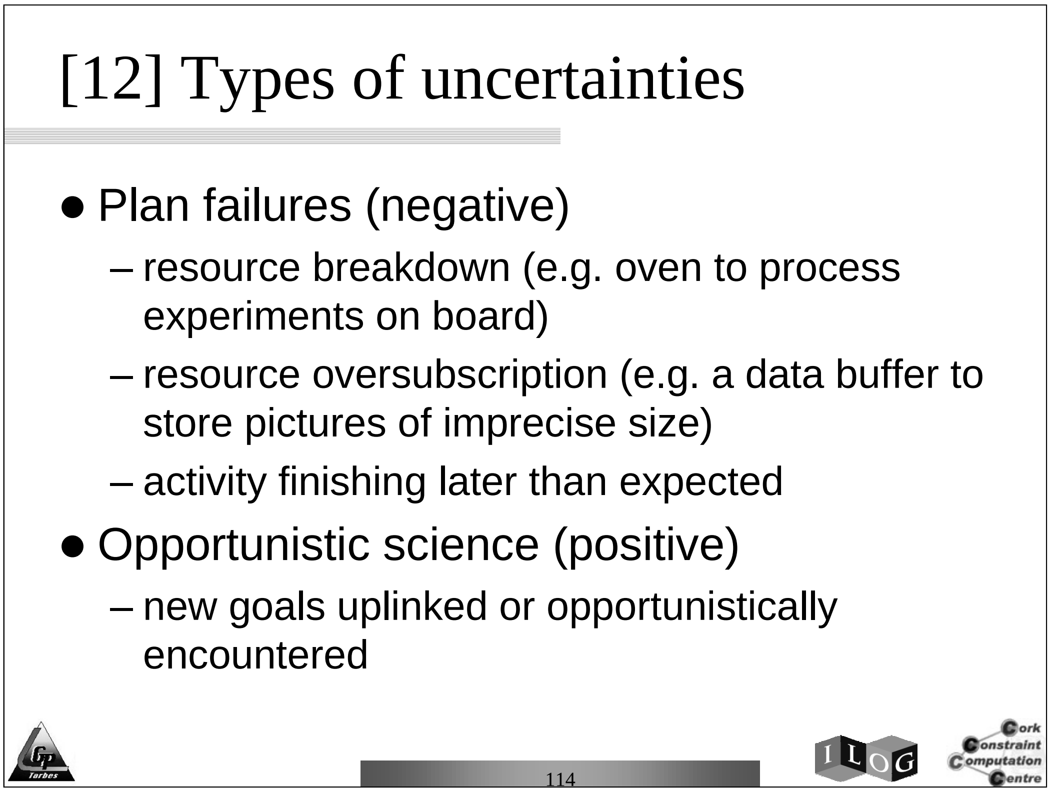
112

13. **[Washington et al. 00]** Washington, R., Golden, K. and Bresina, J. *Plan Execution, Monitoring, and Adaptation for Planetary Rovers.* Electronic Transactions on Artificial Intelligence, Vol. 4 (2000), Section A, p. 3-21. `http://www.ep.liu.se/ej/etai/2000/004/`.

14. **[Wu et al. 99]** Wu, S.D., Byeon, E. & Storer, R.H. *A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling with robustness,* Operations Research, 47(1). 1999.

[12] Continuous planning

- **[Chien et al. 00]**: JPL, NASA
- Context: rover / spacecraft
- Classical approach (Sojourner):
  - planning = ground batch process, uplinked daily
  - upon failure: stop, downlink data, wait for new plan: ↑ time!
- CASPER: planning on board!
  - uplink science requests, downlink results

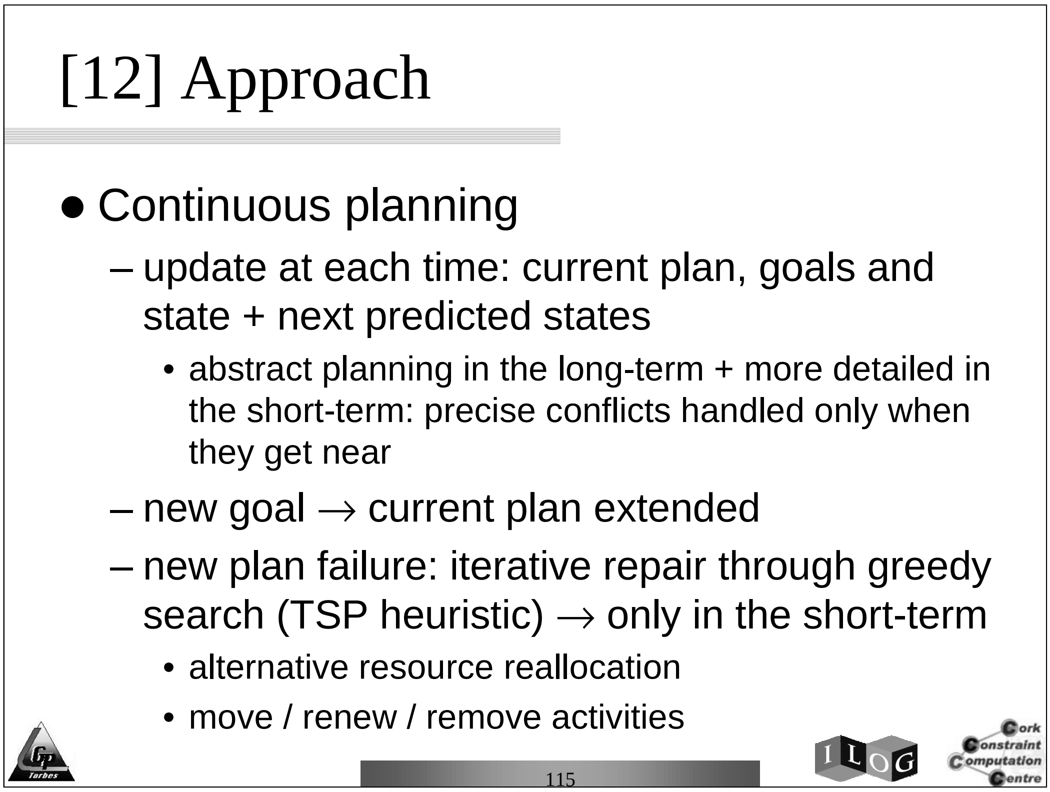


[12] Types of uncertainties

- Plan failures (negative)
  - resource breakdown (e.g. oven to process experiments on board)
  - resource oversubscription (e.g. a data buffer to store pictures of imprecise size)
  - activity finishing later than expected
- Opportunistic science (positive)
  - new goals uplinked or opportunistically encountered

Plan failures = something "bad" occurs: the executed plan will be as good (best case) or less satisfactory than the predicted one.

Opportunistic science = something "good" occurs (actions are faster than expected, some interesting sample detected while the robot has time to take care of it): The executed plan will incorporate more actions and be actually better than the predicted one.



[12] Approach

- Continuous planning
  - – update at each time: current plan, goals and state + next predicted states
    - abstract planning in the long-term + more detailed in the short-term: precise conflicts handled only when they get near
  - – new goal → current plan extended
  - – new plan failure: iterative repair through greedy search (TSP heuristic) → only in the short-term
    - alternative resource reallocation
    - move / renew / remove activities

115

Iterative Repair = local search technique that tries to modify the current solution (here: the plan) locally until it reaches a satisfactory (here: consistent) one. In the best case these techniques minimize changes between the initial failed plan and the final one.

TSP = traveling salesman problem.

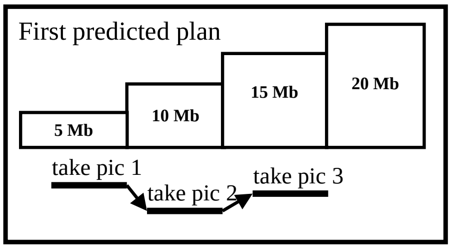
Possible modifications for recovery:

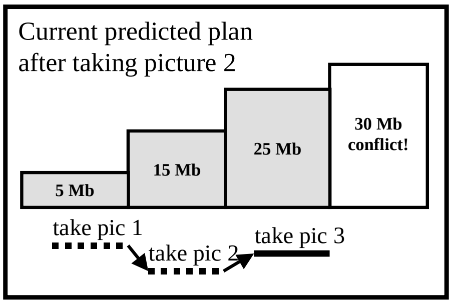
- a resource fails → look for a backup resource that is available and reallocate

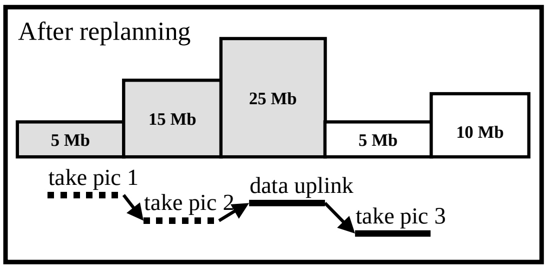
- an action takes longer than expected → postpone further actions

- a shared resource gets oversubscribed (previous actions have used more than expected) → insert an action whose effect is to free the resource (see next slide)

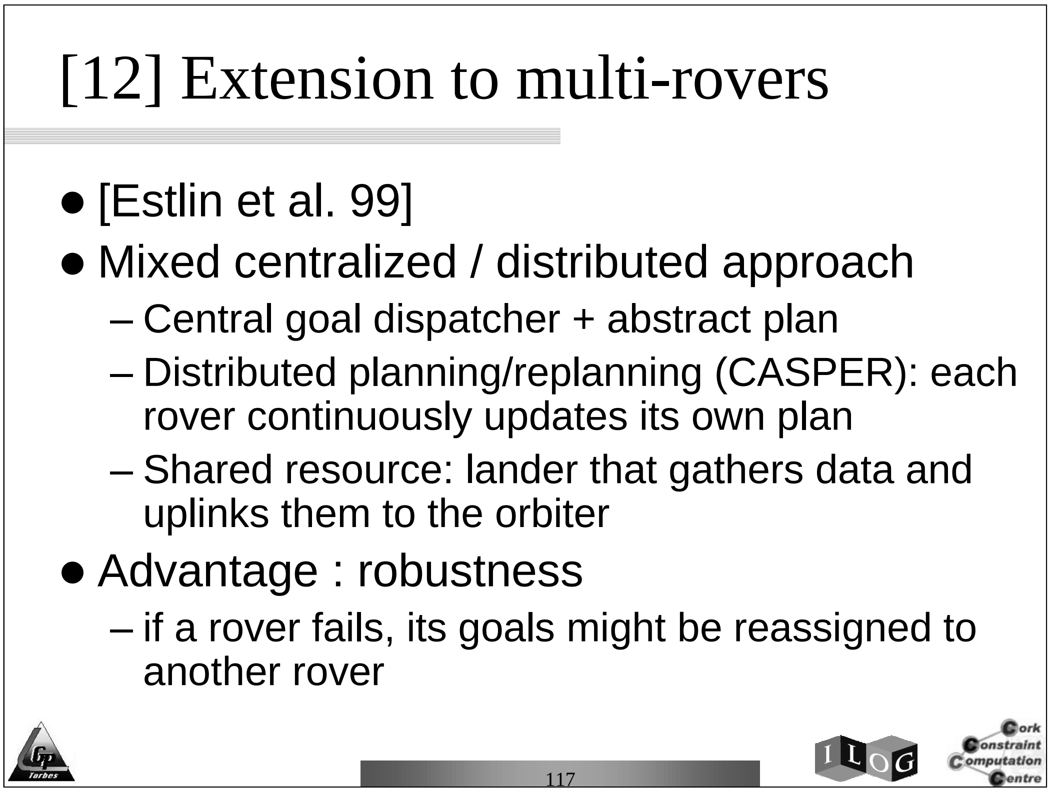
[12] Example


First predicted plan


Current predicted plan after taking picture 2

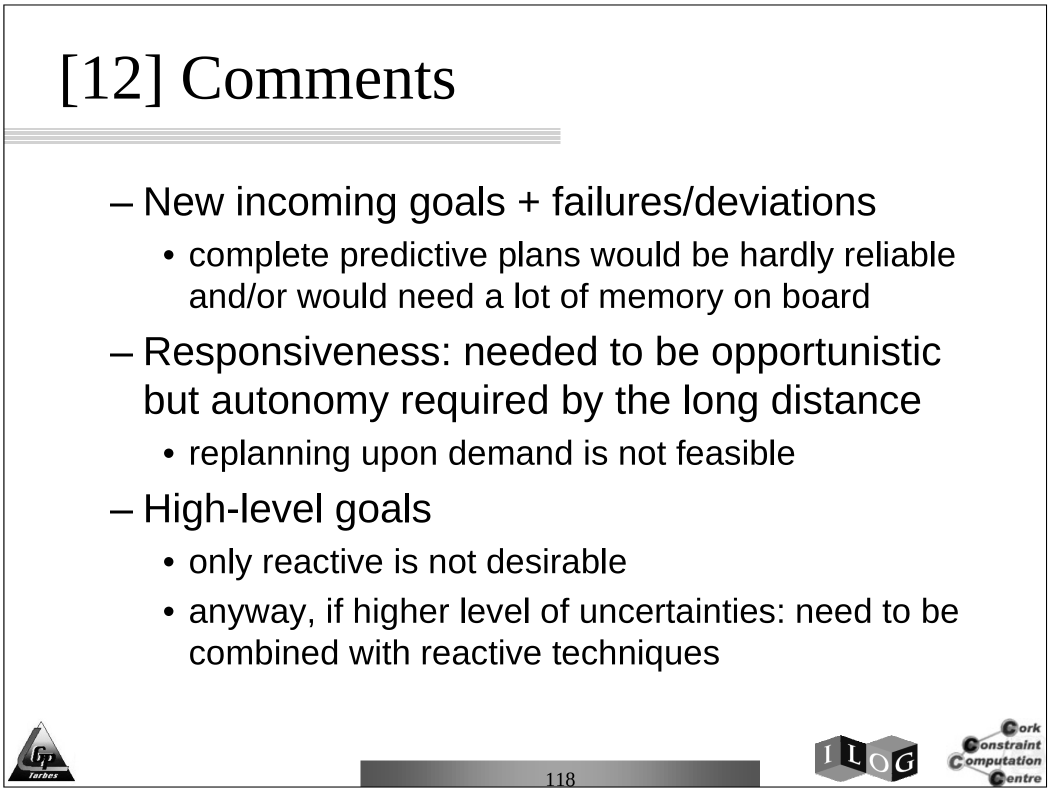

After replanning

- Oversubscription
  - e.g. a data buffer → insert a data uplink action



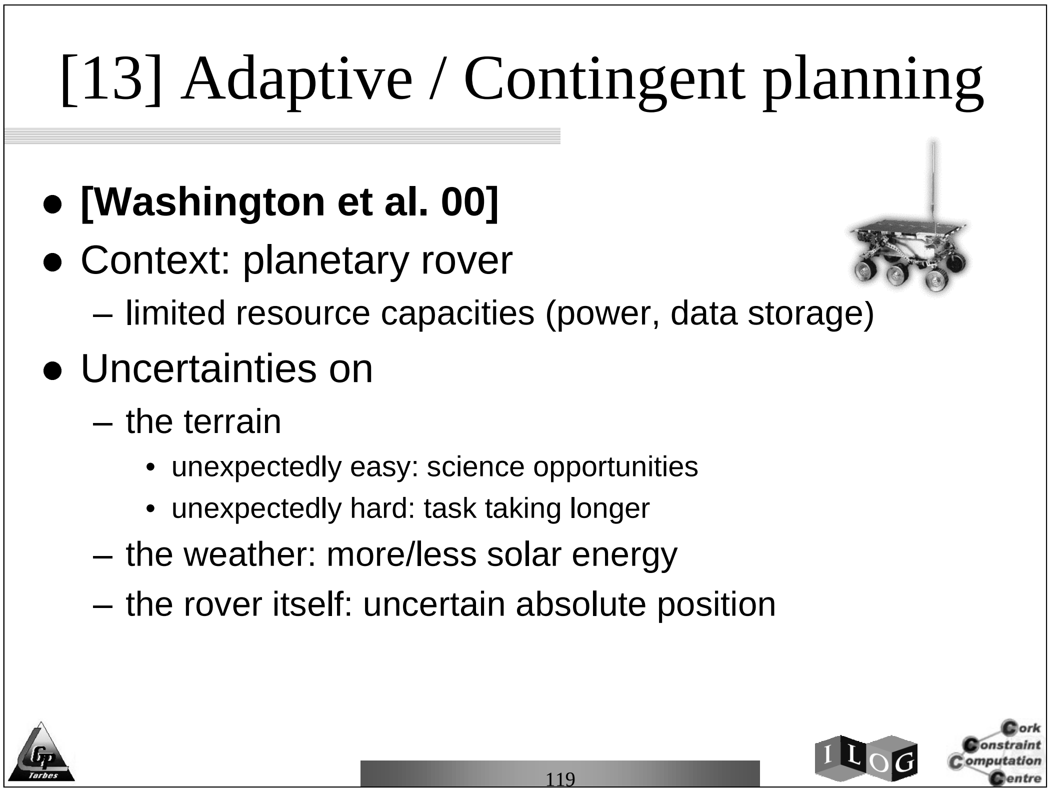
[12] Extension to multi-rovers

- [Estlin et al. 99]
- Mixed centralized / distributed approach
  - Central goal dispatcher + abstract plan
  - Distributed planning/replanning (CASPER): each rover continuously updates its own plan
  - Shared resource: lander that gathers data and uplinks them to the orbiter
- Advantage : robustness
  - if a rover fails, its goals might be reassigned to another rover



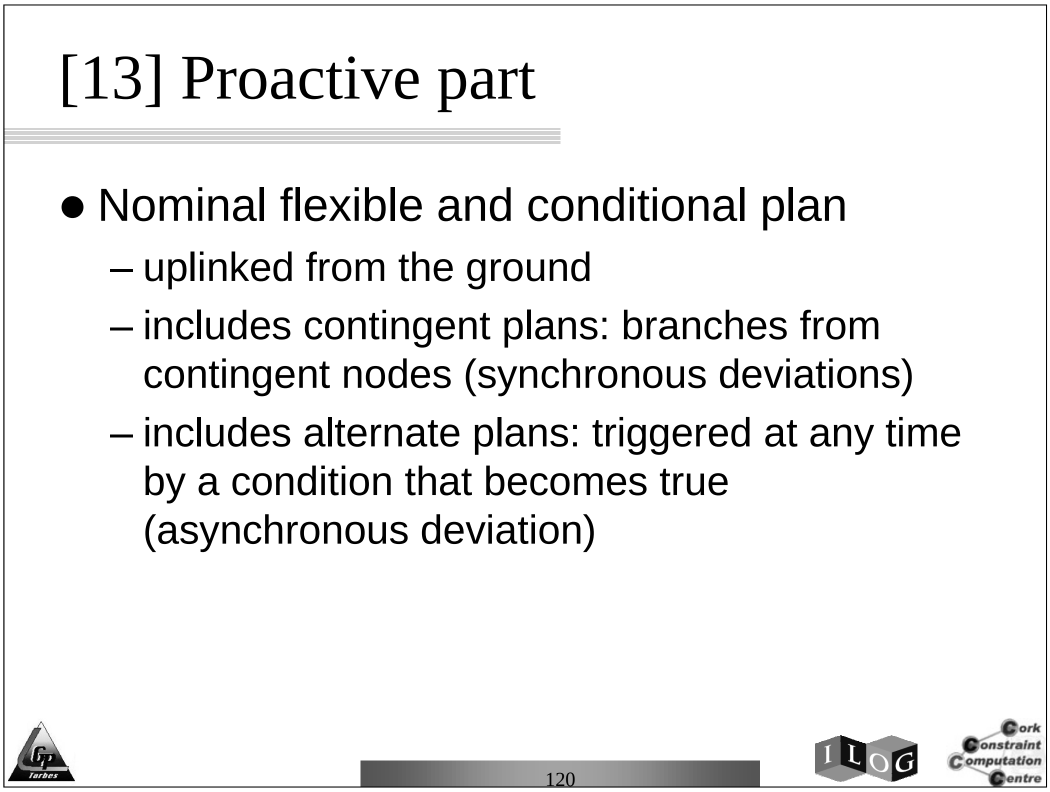
[12] Comments

- New incoming goals + failures/deviations
  - complete predictive plans would be hardly reliable and/or would need a lot of memory on board
- Responsiveness: needed to be opportunistic but autonomy required by the long distance
  - replanning upon demand is not feasible
- High-level goals
  - only reactive is not desirable
  - anyway, if higher level of uncertainties: need to be combined with reactive techniques



[13] Adaptive / Contingent planning

- **[Washington et al. 00]**
- Context: planetary rover
  - – limited resource capacities (power, data storage)
- Uncertainties on
  - – the terrain
    - unexpectedly easy: science opportunities
    - unexpectedly hard: task taking longer
  - – the weather: more/less solar energy
  - – the rover itself: uncertain absolute position



[13] Proactive part

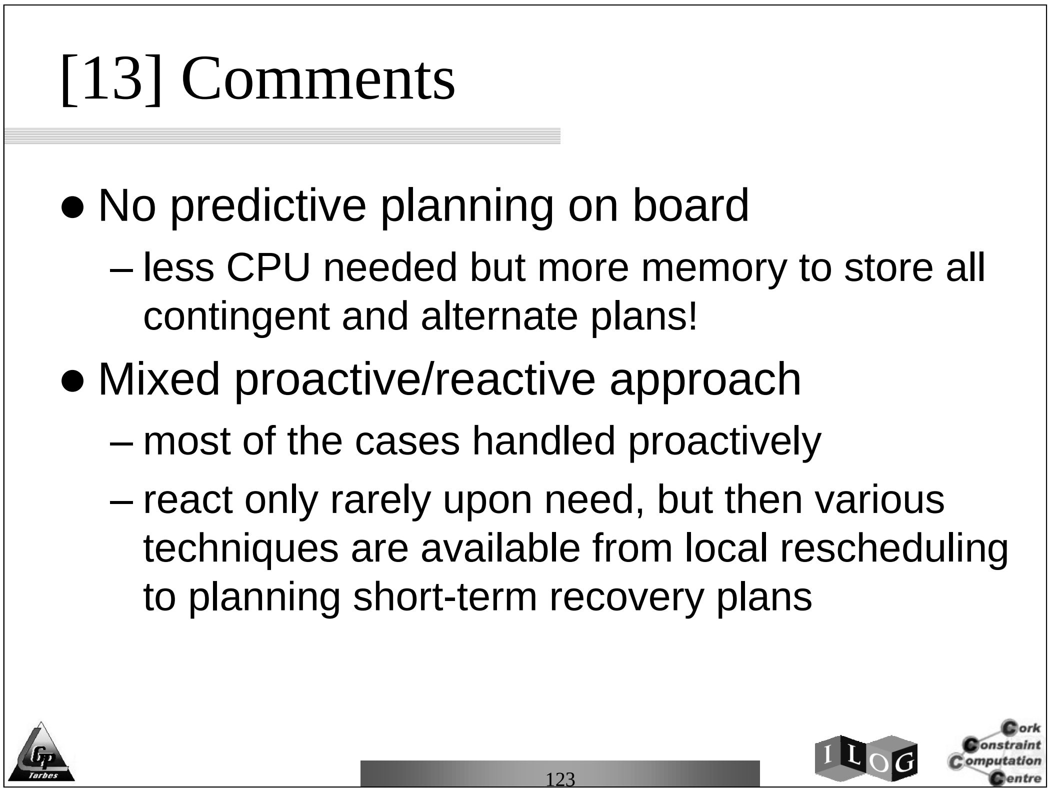
- Nominal flexible and conditional plan
  - uplinked from the ground
  - includes contingent plans: branches from contingent nodes (synchronous deviations)
  - includes alternate plans: triggered at any time by a condition that becomes true (asynchronous deviation)

[13] On-line execution

- Executive:
  - resource manager
  - conflict identification
- Conflict: various possible recoveries
  - a contingent branch at that point solves it
  - start an alternate plan (might be anticipative!)
  - fail → stable state then replan
  - ignore: e.g. too far in the future

[13] Reactive part

- Upon failure
  - recovery plans can be computed by the state identification module
  - the resource manager provides on-board rescheduling capabilities
    - e.g. look for an alternate resource if the allocated one is not available

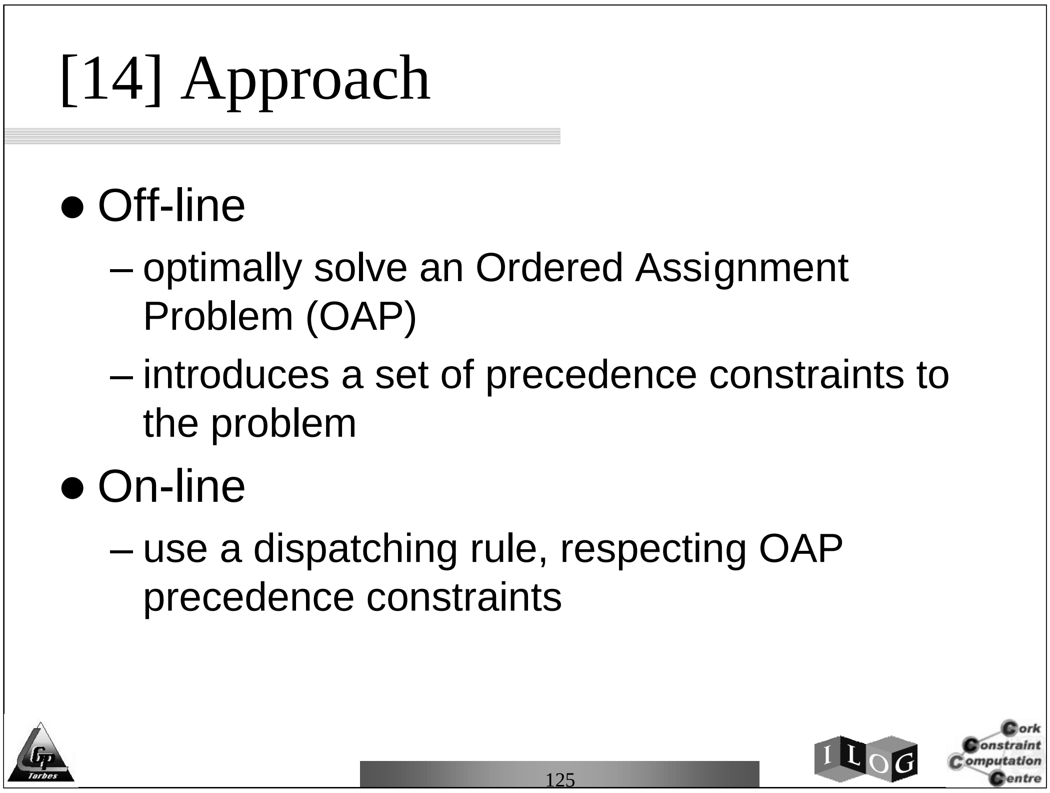


[13] Comments

- No predictive planning on board
  - less CPU needed but more memory to store all contingent and alternate plans!
- Mixed proactive/reactive approach
  - most of the cases handled proactively
  - react only rarely upon need, but then various techniques are available from local rescheduling to planning short-term recovery plans

[14] Off-line partitions

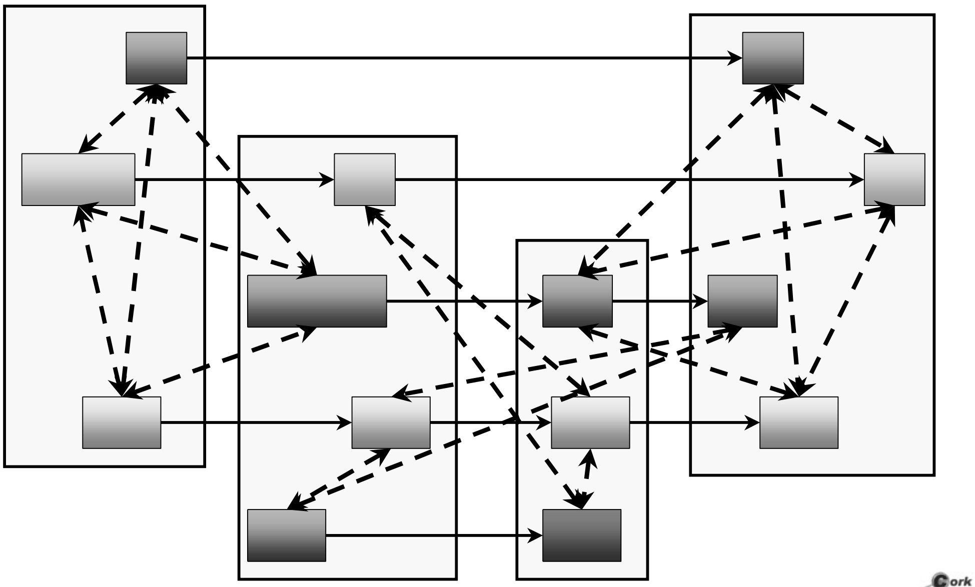
- **[Wu et al. 99]**
- Identify a critical subset of decisions that, to a large extent, dictate global schedule performance
- Make these decisions off-line
- Make the rest of the decisions on-line



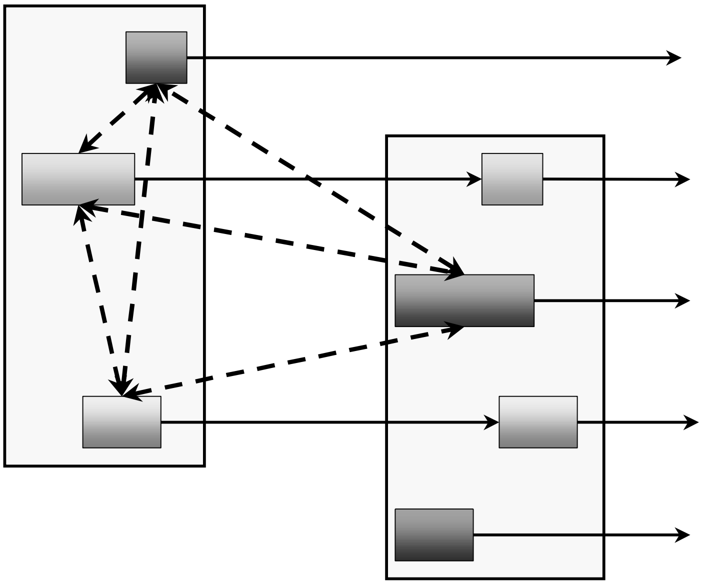
[14] Approach

- Off-line
  - – optimally solve an Ordered Assignment Problem (OAP)
  - – introduces a set of precedence constraints to the problem
- On-line
  - – use a dispatching rule, respecting OAP precedence constraints

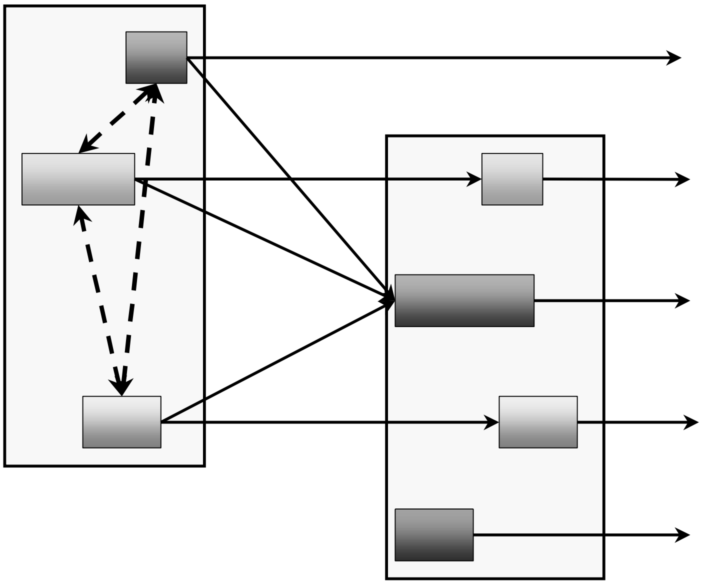
[14] OAP



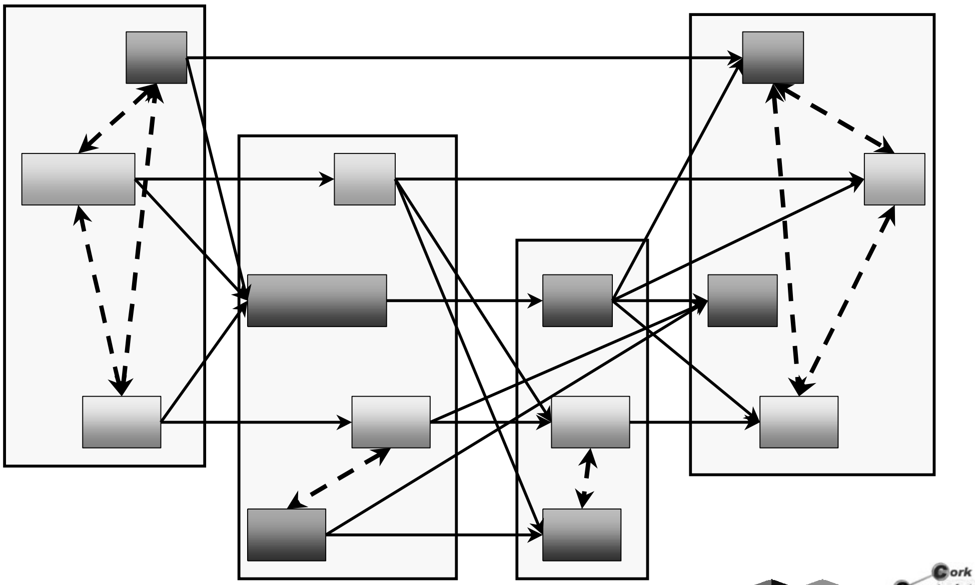
[14] OAP

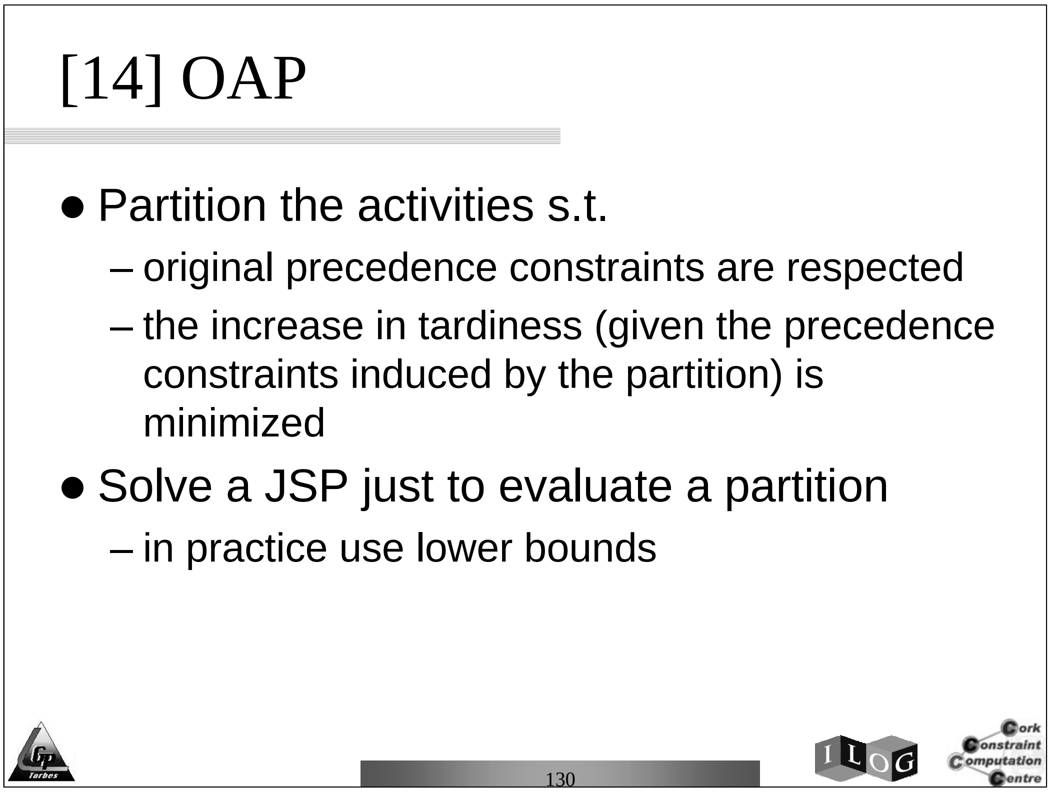


[14] OAP



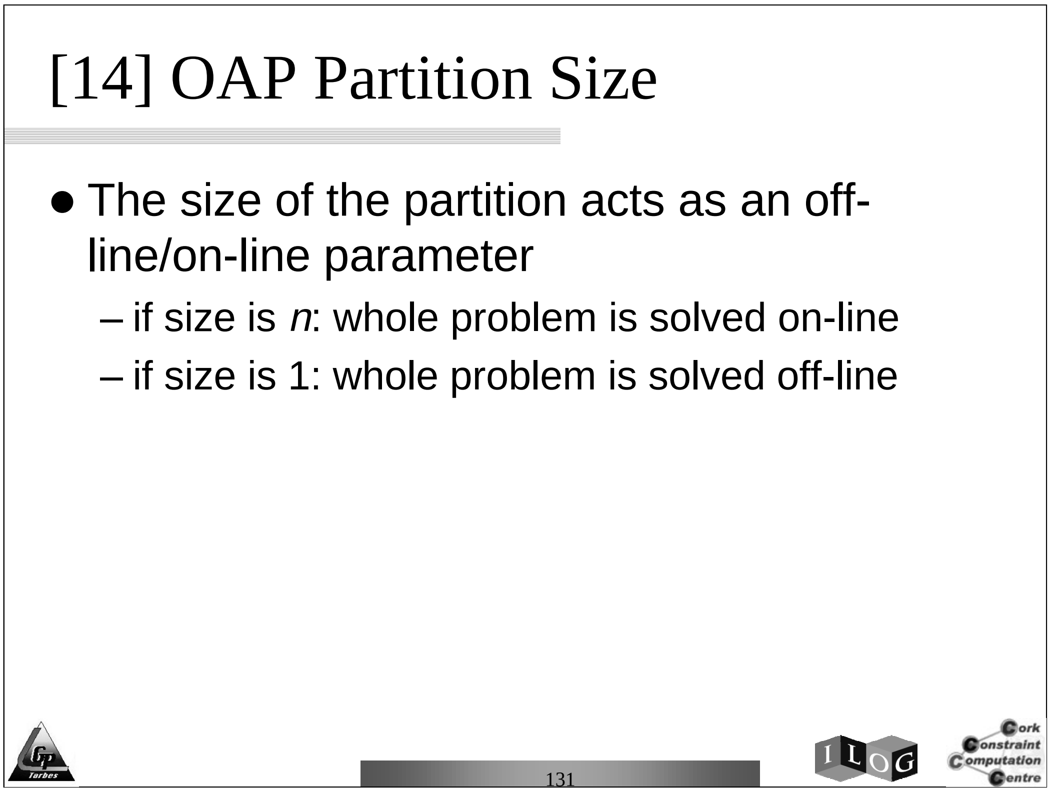
[14] OAP





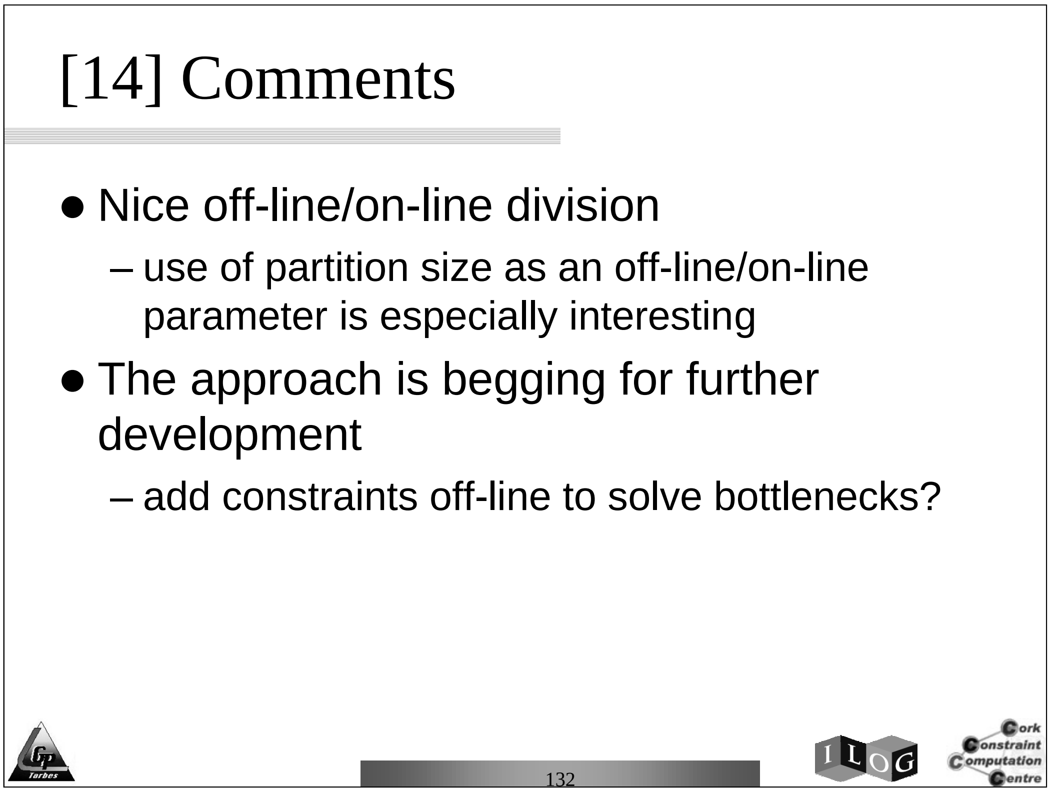
[14] OAP

- Partition the activities s.t.
  - original precedence constraints are respected
  - the increase in tardiness (given the precedence constraints induced by the partition) is minimized
- Solve a JSP just to evaluate a partition
  - in practice use lower bounds



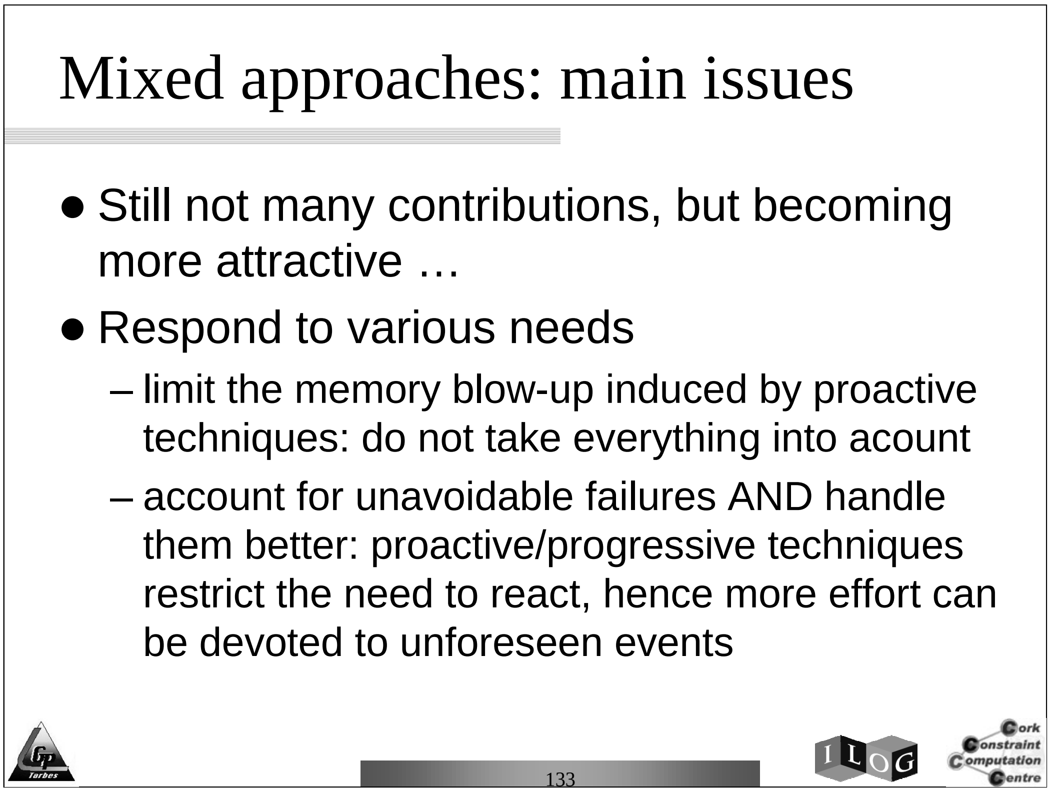
[14] OAP Partition Size

- The size of the partition acts as an off-line/on-line parameter
  - if size is *n*: whole problem is solved on-line
  - if size is 1: whole problem is solved off-line



[14] Comments

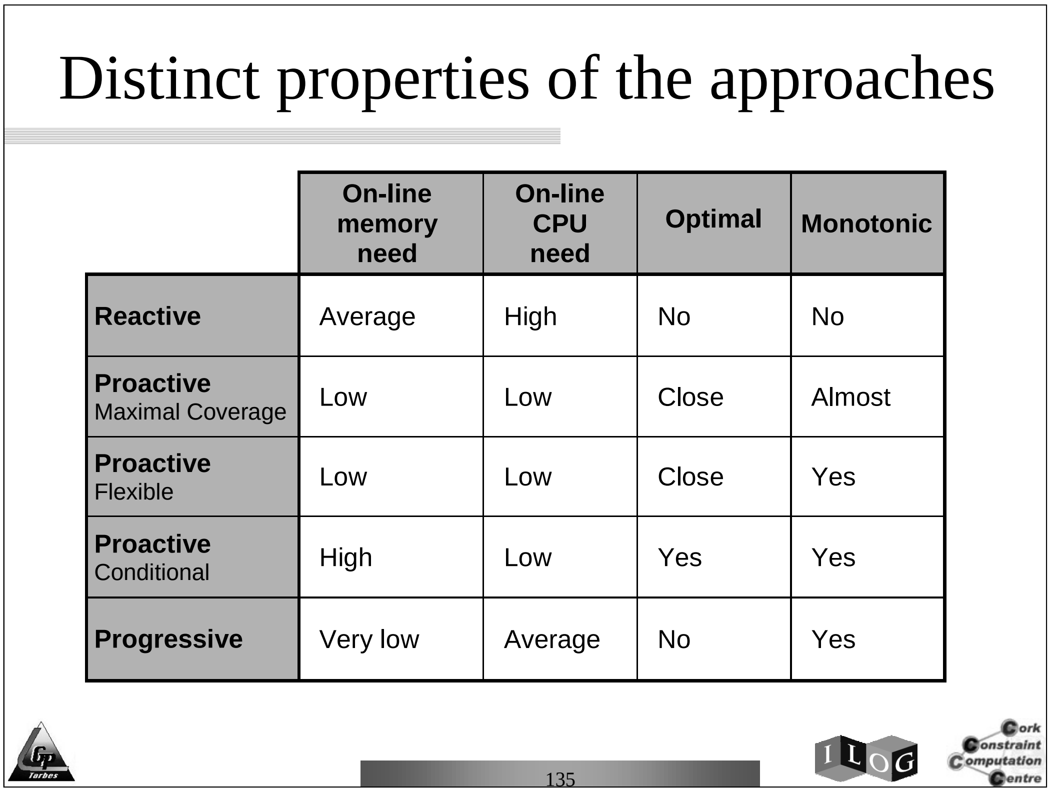
- Nice off-line/on-line division
  - use of partition size as an off-line/on-line parameter is especially interesting
- The approach is begging for further development
  - add constraints off-line to solve bottlenecks?



Mixed approaches: main issues

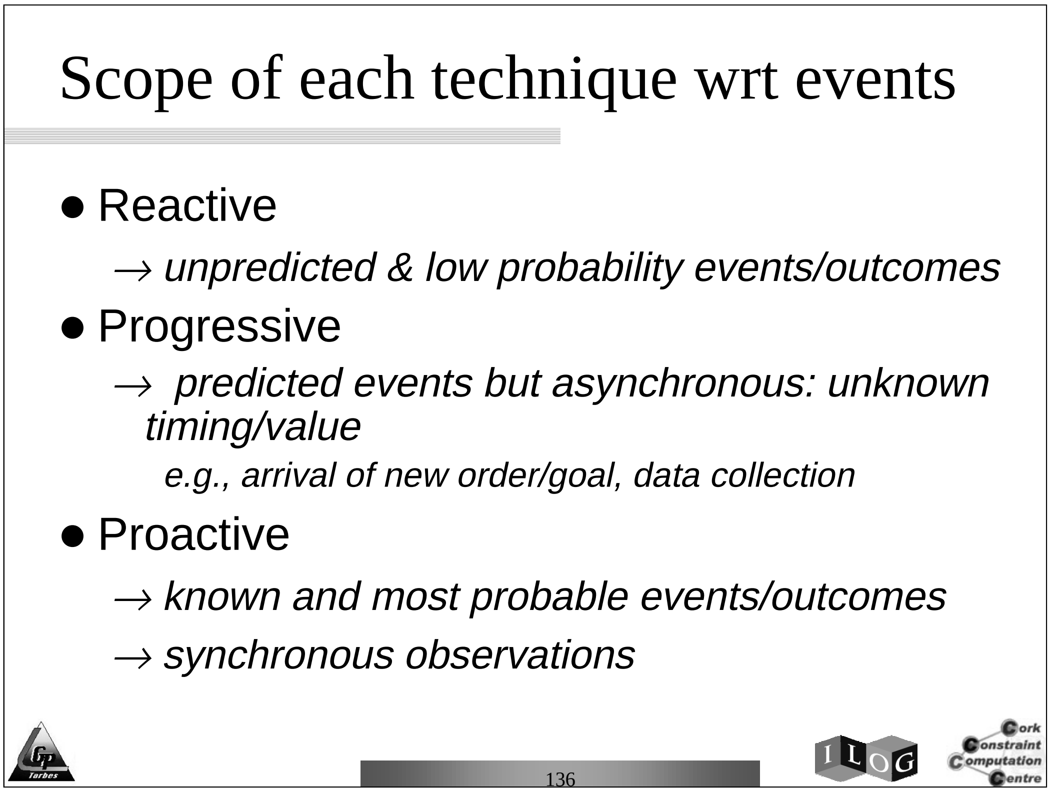
- Still not many contributions, but becoming more attractive …
- Respond to various needs
  - limit the memory blow-up induced by proactive techniques: do not take everything into acount
  - account for unavoidable failures AND handle them better: proactive/progressive techniques restrict the need to react, hence more effort can be devoted to unforeseen events

Summary



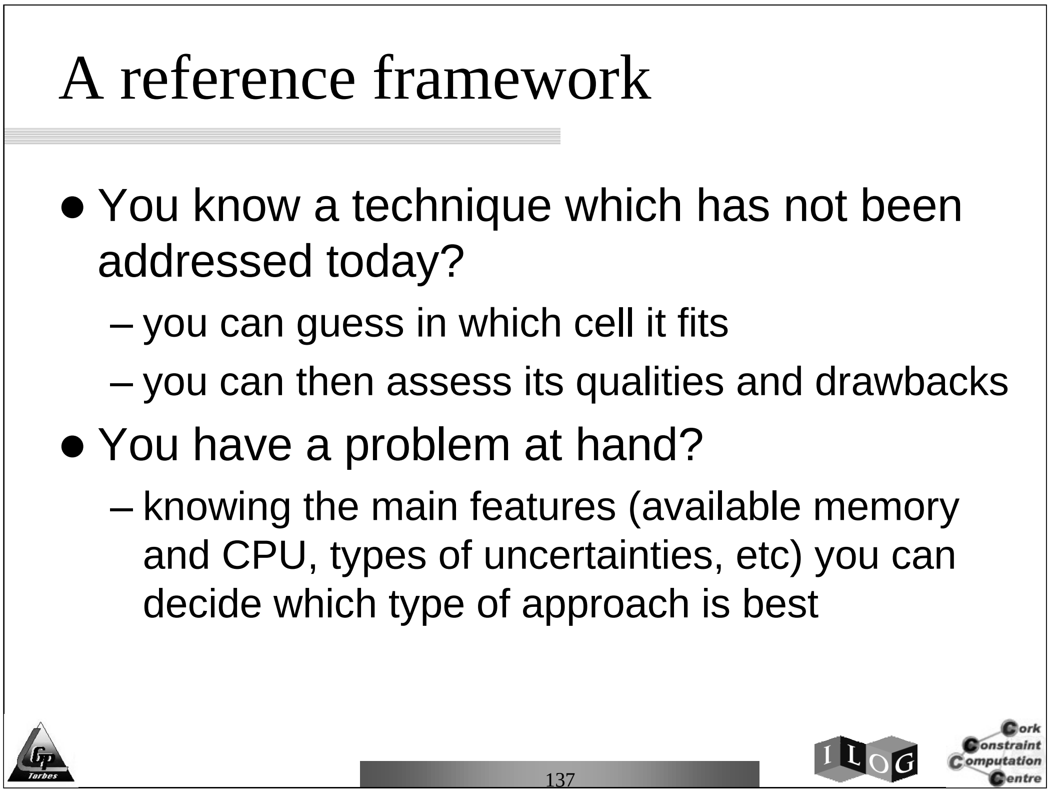
Distinct properties of the approaches

	On-line memory need	On-line CPU need	Optimal	Monotonic
Reactive	Average	High	No	No
Proactive Maximal Coverage	Low	Low	Close	Almost
Proactive Flexible	Low	Low	Close	Yes
Proactive Conditional	High	Low	Yes	Yes
Progressive	Very low	Average	No	Yes



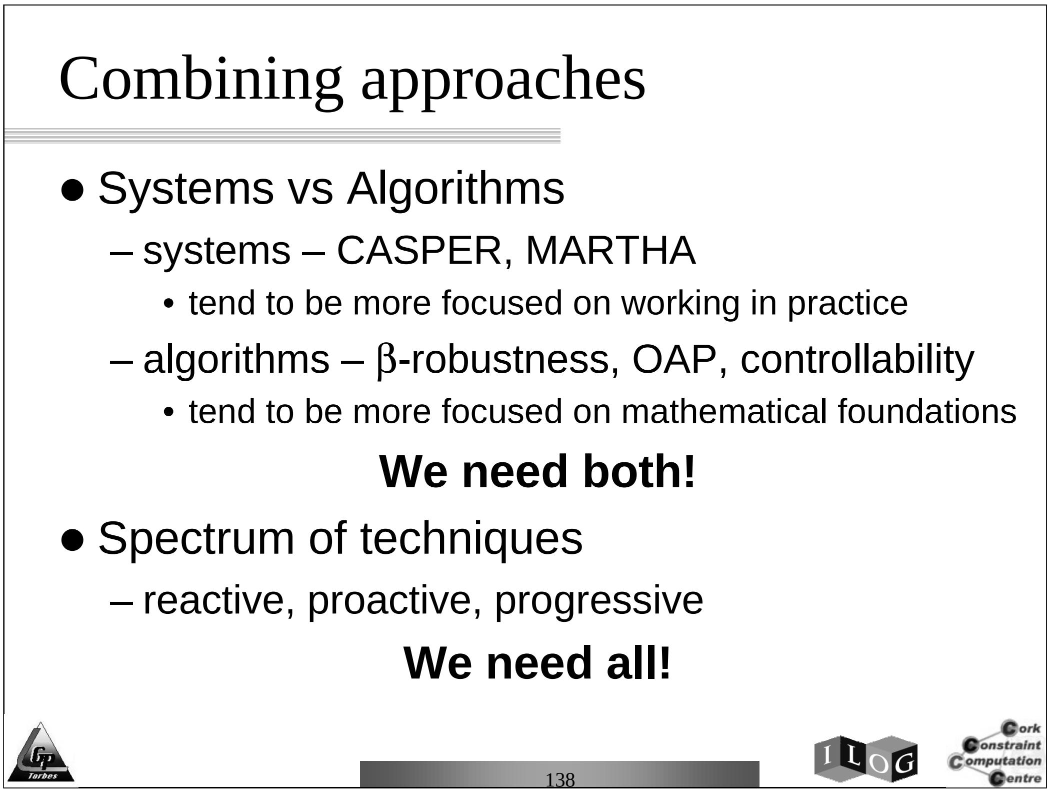
Scope of each technique wrt events

- Reactive
  - → *unpredicted & low probability events/outcomes*
- Progressive
  - → *predicted events but asynchronous: unknown timing/value*
    - *e.g., arrival of new order/goal, data collection*
- Proactive
  - → *known and most probable events/outcomes*
  - → *synchronous observations*



A reference framework

- You know a technique which has not been addressed today?
  - you can guess in which cell it fits
  - you can then assess its qualities and drawbacks
- You have a problem at hand?
  - knowing the main features (available memory and CPU, types of uncertainties, etc) you can decide which type of approach is best



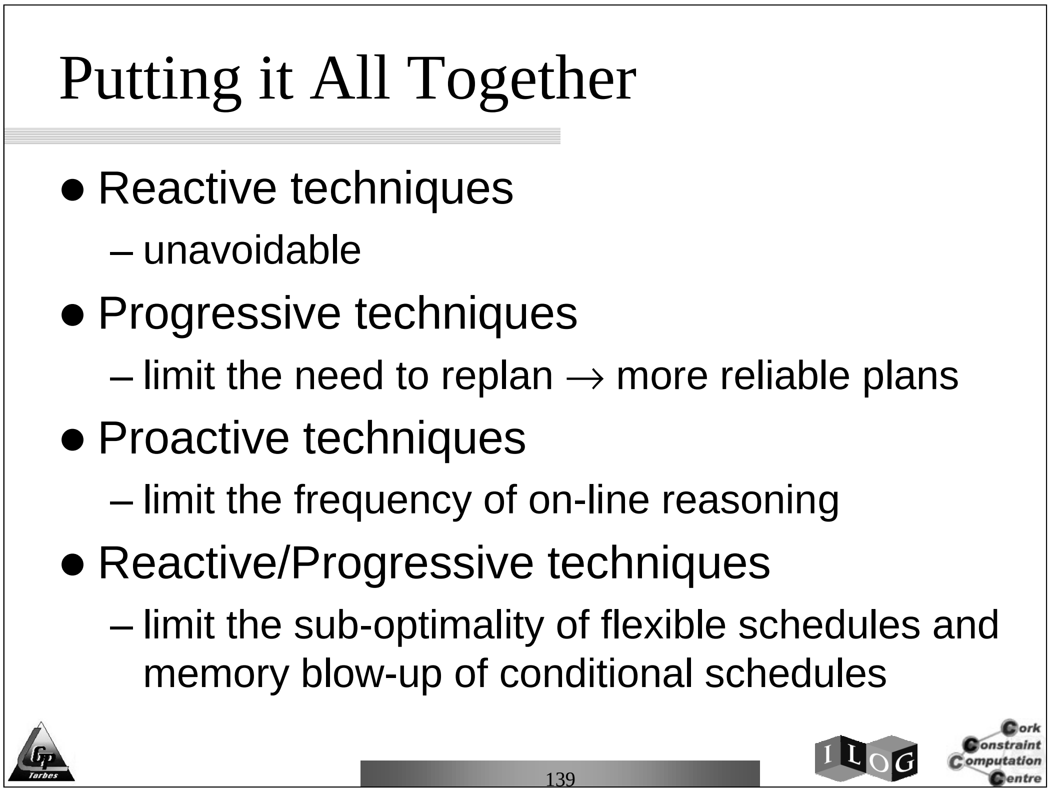
Combining approaches

- Systems vs Algorithms
  - systems – CASPER, MARTHA
    - tend to be more focused on working in practice
  - algorithms – β-robustness, OAP, controllability
    - tend to be more focused on mathematical foundations

We need both!

- Spectrum of techniques
  - reactive, proactive, progressive

We need all!



Putting it All Together

- Reactive techniques
  - unavoidable
- Progressive techniques
  - limit the need to replan → more reliable plans
- Proactive techniques
  - limit the frequency of on-line reasoning
- Reactive/Progressive techniques
  - limit the sub-optimality of flexible schedules and memory blow-up of conditional schedules

Questions?