Planning & Scheduling with Over-subscribed Resources, Preferences, and Soft Constraints

Minh Do, Terry Zimmerman, and
Subbarao Kambhampati

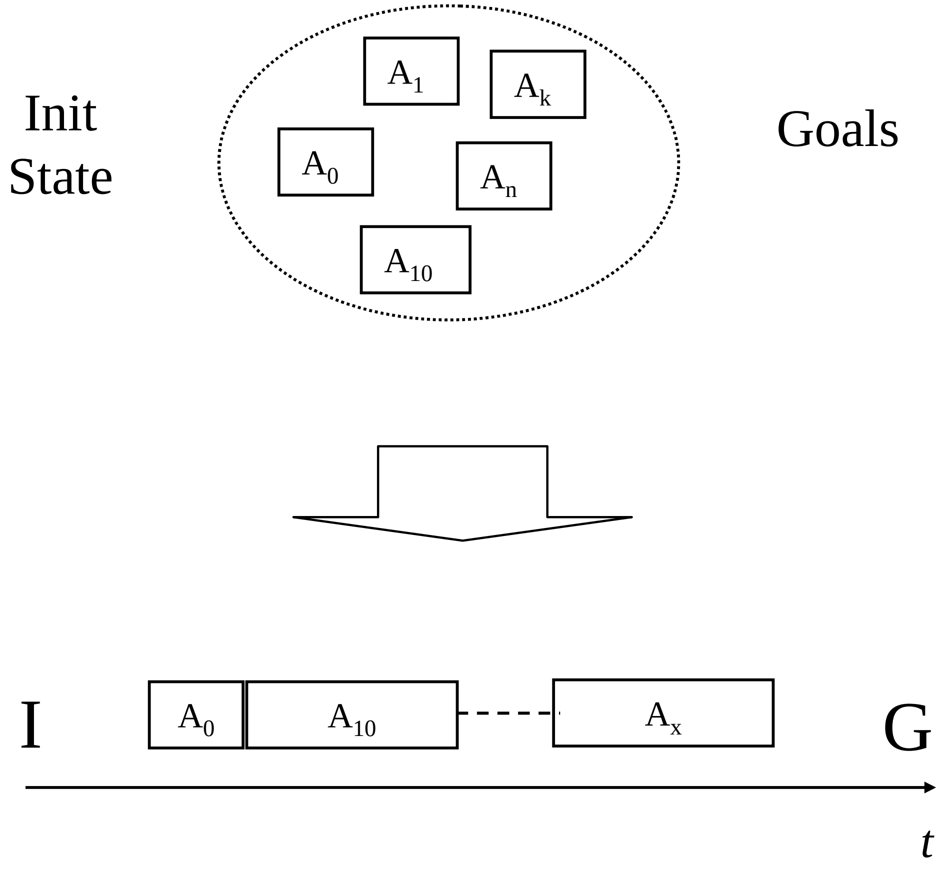
parc Palo Alto Research Center

Carnegie Mellon THE ROBOTICS INSTITUTE

ASU

Planning & Scheduling: Traditional View

planning



Init State

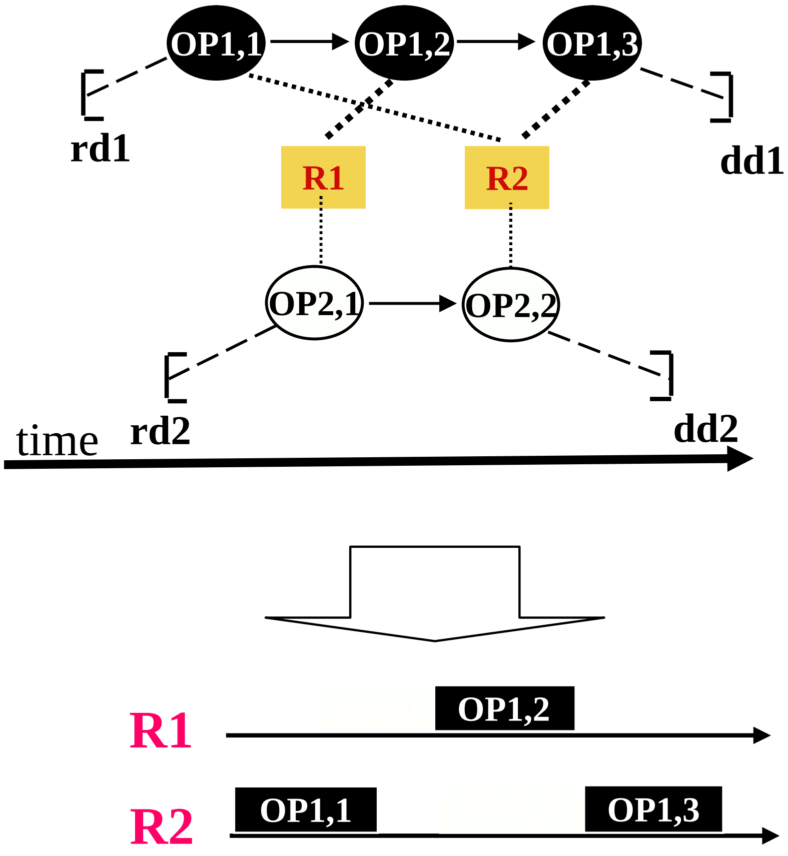
A_1 A_k A_0 A_n A_{10}

Goals

I A_0 A_{10} A_x G

t

scheduling



OP1,1 OP1,2 OP1,3

rd1 R1 R2 dd1

OP2,1 OP2,2

time rd2 dd2

R1 OP1,2

R2 OP1,1 OP1,3

Oversubscription in Planning & Scheduling: Sources

planning

User: achieve all goals (at the lowest cost)!

Planner:

- Not enough resources
- Conflicting goals
- Goal achievement cost outweighs goal utility

Over-subscribed/Partial Satisfaction

(i.e. achieve the **"best"** subset of goals)

scheduling

User: allocate all tasks (maximize objective)!

Scheduler:

- Not enough resources (e.g. time, aircraft, fuel)
- Conflicting tasks

Over-subscribed

(i.e. allocate the **"best"** subset of tasks)

Oversubscription: best solution does not need to satisfy all (soft) constraints

Oversubscription in Planning & Scheduling: Representation

Solution: Soft Constraints + Preferences

Planning Preferences

1. Over (sub)set of Goals to achieve
2. Over plan trajectory
(i.e. actions selected, states visisted)
3. Over plan stability
(i.e. changes from prev plan)

Scheduling Preferences

1. Over (sub)set of tasks to allocate
2. Over set of resource constraints
3. Over schedule robustness and stability

Optimizing according to objective, preferences, priority, with penalty for violating soft constraints (*must satisfy all hard constraints*)

Applications/Examples

- Mars Rover attempting to maximize scientific return with limited resources (Smith, 2004)
- UAVs attempting to maximize reconnaissance returns, given fuel etc constraints
- Logistics problems with time and resource constraints
- Manufacturing with multiple job requests with deadlines (Ruml et. al., 2005)
- Shopping recommendation system with user's preferences on product features (Brafman, 2003)

- NASA's shuttle ground processing (Deale et al., 1994)
- Telescope observations scheduling (Bresina, 1996), Hubble space telescope observations (Johnston, M., and Miller, G.)1994.
- Satellite observation scheduling (Frank, Jonsson, Morris, & Smith, 2001; Globus, Crawford, Lohn, & Pryor, 2003)
- USAF Air Mobility Command (AMC) airlift (Kramer & Smith, 2003)
- AFSCN -multiple resource satellite communications (Barbulescu, Howe, Roberts 2006)
- *more to come….*

Is this really a *different* problem?

Skeptics Corner

Once we have optimization, we have soft constraints and preferences already

- » Prefer shortest makespan, highest slack, lowest tardiness plan/schedule

..So the distinction between normal and over-subscription based on soft constraints seems somewhat fuzzy..

- » Particularly so for scheduling which always had optimization issues front and center

We can however make the demarcation clear by noting that we are interested in *problem specific* (rather than global, problem independent) soft constraints

- » E.g. Having both right and left shoes together is significantly more preferable to having either one.

Tutorial Outline

- Introduction (15)
- Oversubscription in Scheduling: Basic view (Terry) (75m)

(Break)

- Oversubscription Planning: Basic view (Minh)(75min)
- Extensions for Scheduling: Reasoning over quality metrics (Terry) (20m)
- Extensions for Planning: Metric goals, goal dependencies etc. (Minh)(20m)

Scheduling

Tutorial Outline

- Introduction (15)
- Oversubscription in Scheduling: Basic view (Terry) (75m)

(Break)

- Oversubscription Planning: Basic view (Minh)(75min)
- Extensions for Scheduling: Reasoning over quality metrics (Terry) (20m)
- Extensions for Planning: Metric goals, goal dependencies etc. (Minh)(20m)

Planning: Outline

- Background
- Qualitative Preferences
- Quantitative Preferences
- Extensions

Types of Preferences & Soft Constraints

- Goals
- Actions
- Plan-trajectory

Solving Approaches:

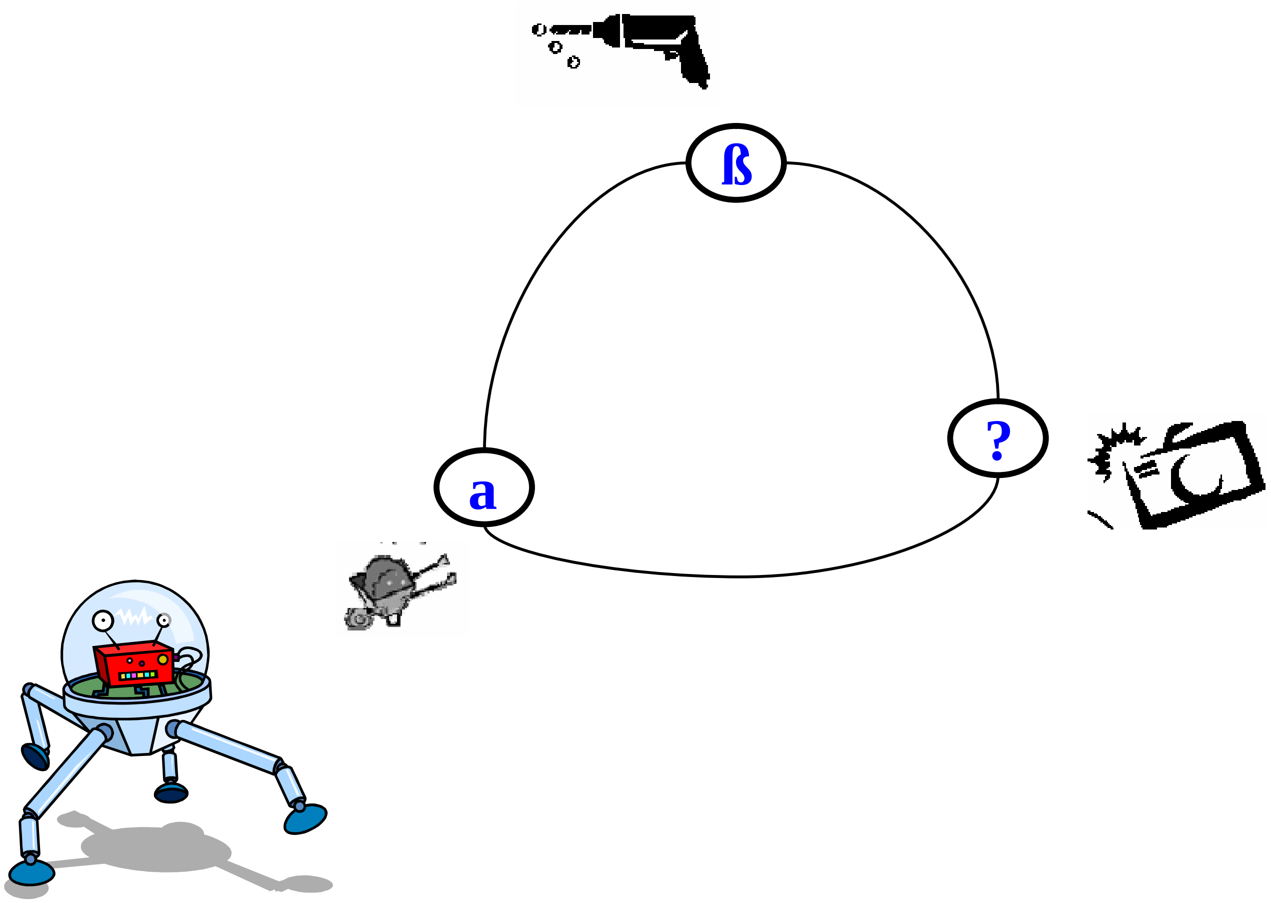
- CSP/ILP/SAT compilation
- Heuristic search
- Compiling away preferences & soft constraints

Leading Example: Rover

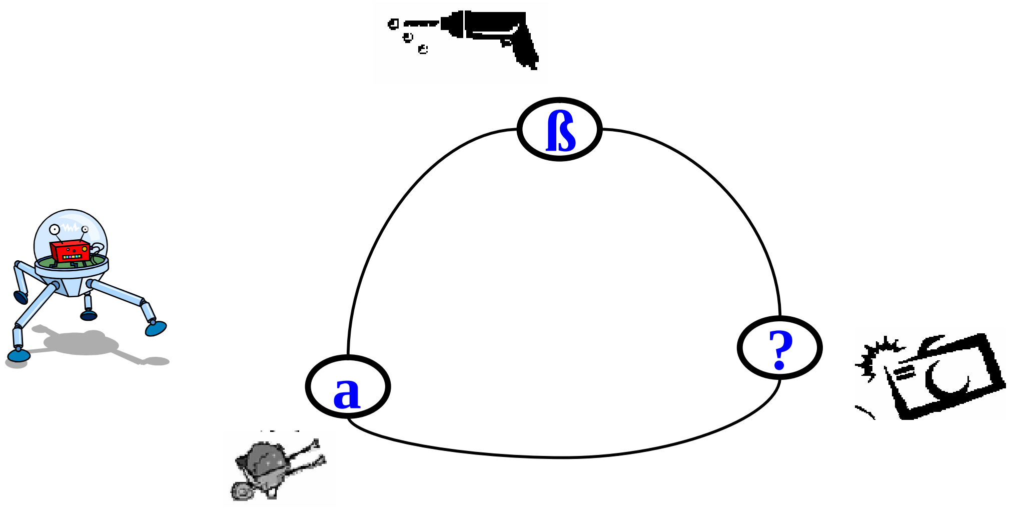
[Bryce & Kambhampati, 2006]

Actions:

Move(a,ß)
Sample(Soil,a)
Sample(Rock,ß)
Take(Picture,?)

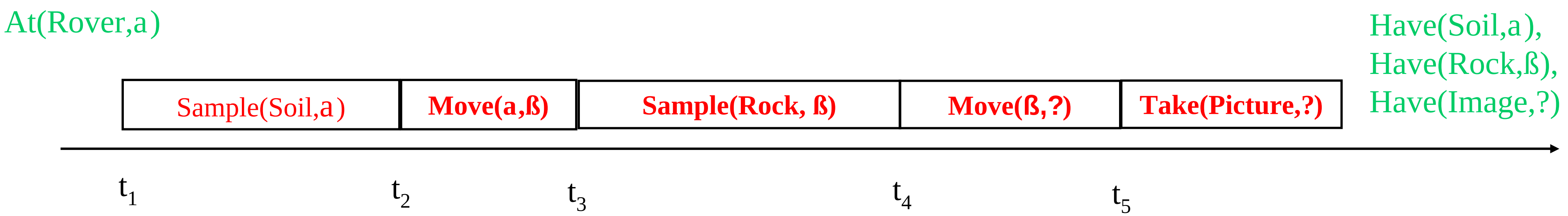


Planning Problem



- Planning Problem in STRIPS:
  - Domain:
    - » Set of binary literals representing world state
      - At(Rover,a), HaveImage(?)
    - » Actions: preconditions ? effects
      - Move(a,ß): At(Rover,a) ? At(Rover,ß)
  - Initial state: fully specified
    - » At(Rover,a), Available(Soil,a), Available(Rock,ß), Visible(Image,?)
  - Goal state: partially specified
    - » Have(Soil), Have(Rock), Have(Image)

Plan



At(Rover,a)

Sample(Soil,a)

Move(a,ß)

Sample(Rock, ß)

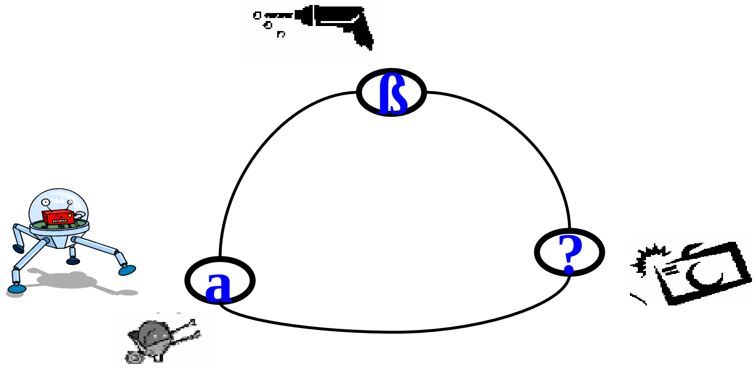
Move(ß,?)

Take(Picture,?)

Have(Soil,a),
Have(Rock,ß),
Have(Image,?)

Set of actions with associated fixed starting time.

? When executed in the initial state will achieve ***all*** goals (at the end)



Oversubscription

? Not enough battery power, time to collect *all* 3 goals
Find the **best** plan within the resource limit?

Preference Representation

How to define **best** plan?

Qualitative Preferences:

Compare plans based on logical preferences between achieved goals and/or plan trajectories

- Simpler, easier to get/elicit
- Less expressive, inadequate for some domains
  - ➢ Many incomparable plans
  - ➢ No notion of "*how much better*" one plan is to another ?

Best plan: not preferred/dominated by any other plan (many of them)

Quantitative Preferences:

compare plans based on real cost/utility values associated with goals actions or constraints on plan trajectories

- More expressive, less number of incomparable plans
- Hard to get exact/approximate "utility" value in many domains

Best plan: plan with highest total value (one or few)

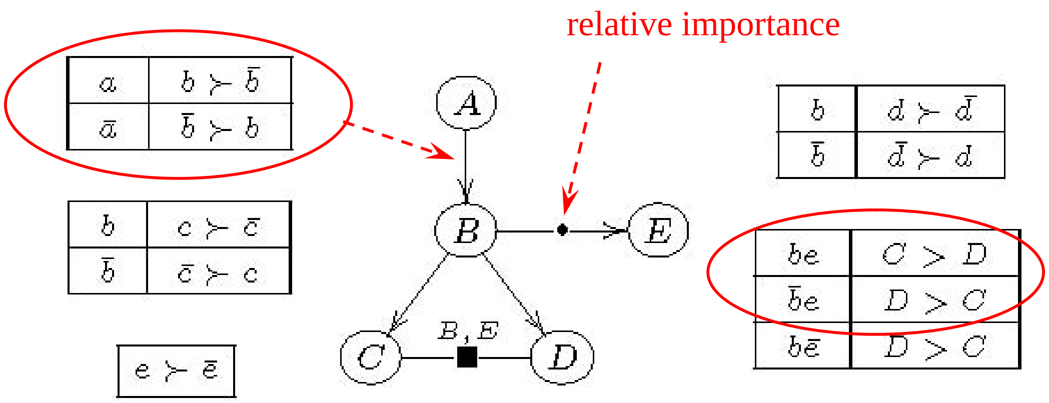
Planning: Outline

- Background
- ➡ Qualitative Preferences
  - Goal preferences in CP-Net
    - CSP search
  - Trajectory preferences in PP
    - S-Models, heuristic search
- Quantitative Preferences
- Extensions

Goal Preferences in TCP-Net

TCP-Net

[Brafman & Chernyavsky, 2005]



a	$b \succ \bar{b}$
$\bar{a}$	$\bar{b} \succ b$

b	$c \succ \bar{c}$
$\bar{b}$	$\bar{c} \succ c$

$e \succ \bar{e}$

b	$d \succ \bar{d}$
$\bar{b}$	$\bar{d} \succ d$

be	$C > D$
$\bar{b}e$	$D > C$
$b\bar{e}$	$D > C$

Also suitable for Multi-Value Planning Representation (SAS+)

Nodes: Goals

Edges:

- Relative importance
- Preference dependencies

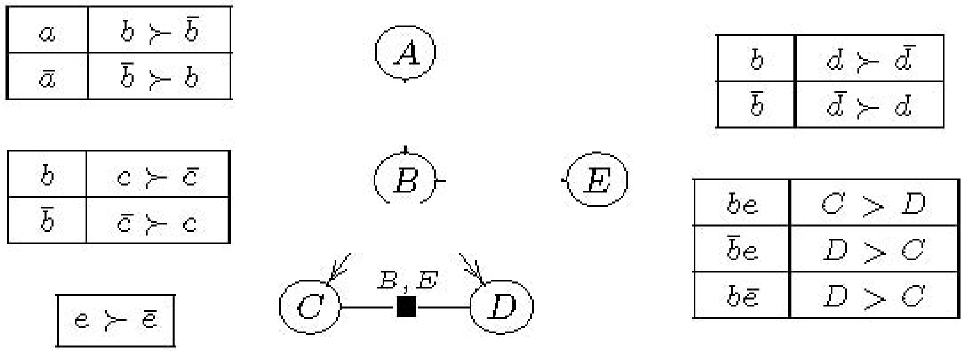
Partial order over set of goal assignments

P1: $\bar{a}\bar{b}\bar{c}de$ $\succ$ P2: $\bar{a}\bar{b}cde$

Objective: find plan *not dominated* by any other plan

Qualitative Preferences: Reasoning

CSP-Based Planner [Brafman & Chernyavsky, 2005]



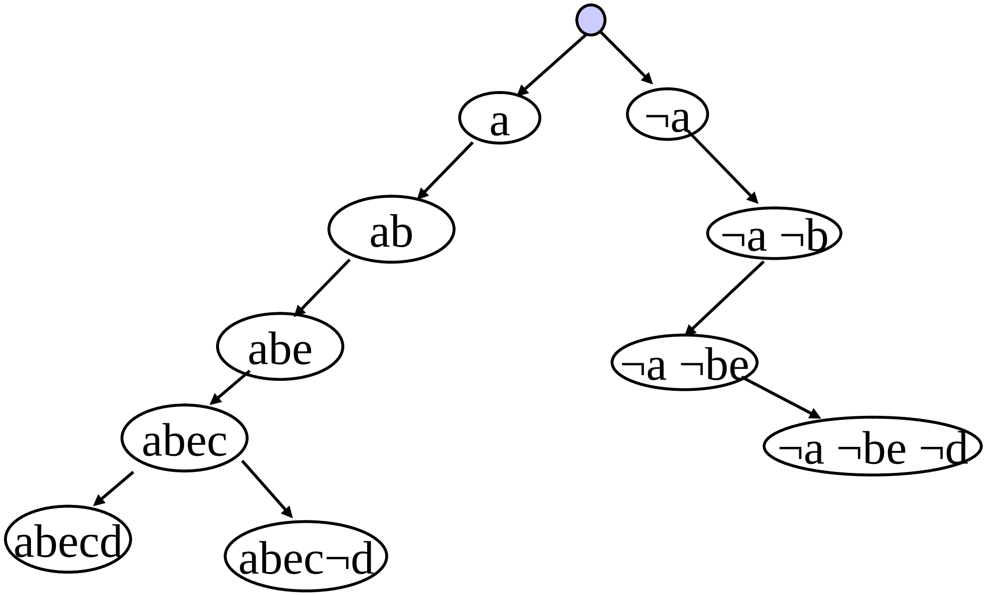
a	$b \succ \bar{b}$
$\bar{a}$	$\bar{b} \succ b$

b	$c \succ \bar{c}$
$\bar{b}$	$\bar{c} \succ c$

$e \succ \bar{e}$

b	$d \succ \bar{d}$
$\bar{b}$	$\bar{d} \succ d$

be	$C > D$
$\bar{b}e$	$D > C$
$b\bar{e}$	$D > C$



Variable Ordering:

Variable with no parent first

A ?   B ?   E ?   C, D

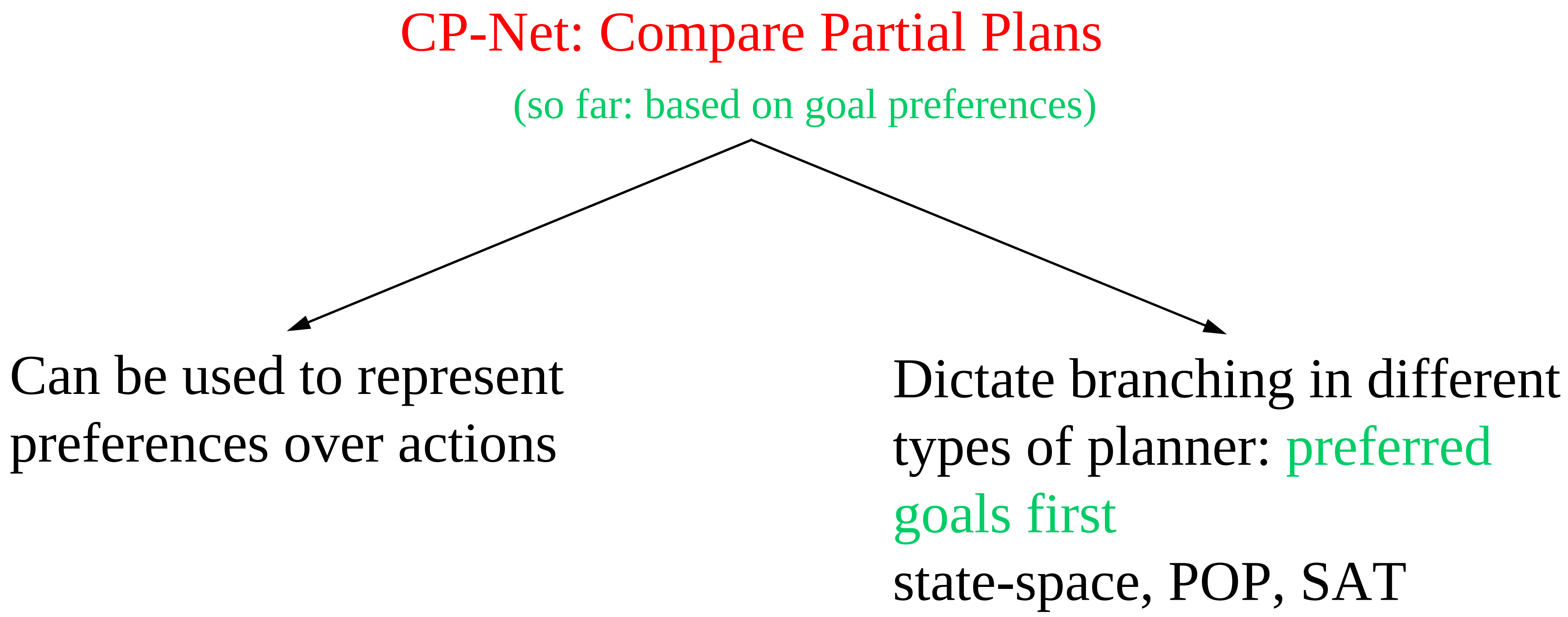
Value Ordering:

Preferred values first (based on assignments to parents)

TCP-Net Update:

Remove assigned variables & edges from TCP-net

CP-Net: Possible Extensions



CP-Net: Compare Partial Plans

(so far: based on goal preferences)

Can be used to represent preferences over actions

Dictate branching in different types of planner: preferred goals first

state-space, POP, SAT

Trajectory Preferences using Logic Programming

[Tran & Pontelli, 2003; Bienvenu et. al., 2006]

PP preference language:

- Preferences on *states* visited by plan trajectory
- Preferences on *actions* occurrences
- Preferences over *trajectory property*
- Using subset of **Temporal Logic** for trajectory preferences (*next*, *always*, *until, eventually*)
- Preferences construction: basic desire ? atomic preference ? general preferences

» Partial order between trajectories based on general preferences

Trajectory Preference: Basic Desire

Basic desire: $\vee, \wedge, \neg, \mathit{next}, \mathit{always}, \mathit{until}, \mathit{eventually}$

over fluent formulae or *occ(a)* (nesting is allowed)

$\varphi_1 = \mathit{next}(\mathit{at}(\beta))$ $\quad$ $\varphi_2 = \mathit{eventually}(\mathit{occ}(\mathit{CollectSoil}(\gamma)))$

$\varphi_3 = \mathit{goal}(\mathit{Have}(\mathit{Soil}) \wedge \mathit{Have}(\mathit{Rock})) \vee \mathit{Have}(\mathrm{Image}))$

trajectory comparison

$$\alpha \prec_\varphi \beta : \quad (\alpha \mapsto \varphi) \wedge (\neg(\beta \mapsto \varphi))$$

$$\alpha \approx_\varphi \beta : \quad \neg(\alpha \prec_\varphi \beta) \wedge \neg(\beta \prec_\varphi \alpha)$$

Trajectory Preference: Atomic Preference

Atomic Preference: Preferences over basic desires

$\varphi_1 = next(at(\beta))$ $\varphi_2 = eventually(occ(CollectSoil(\gamma)))$

$\varphi_3 = goal(Have(Soil) \wedge Have(Rock)) \vee Have(\text{Image}))$

$$\psi_1 = \psi_1 \lhd \psi_2 \lhd \psi_3$$

trajectory comparison

$\alpha \prec_\psi \beta : \quad \exists i : \forall j < i : \alpha \approx_{\varphi_j} \beta, \; \alpha \prec_{\varphi_i} \beta$

$\alpha \approx_\psi \beta : \quad \neg(\alpha \prec_\psi \beta) \wedge \neg(\beta \prec_\psi \alpha)$

Trajectory Preference: General Preferences

$\varphi_1 = next(at(\beta))$ $\varphi_2 = eventually(occ(CollectSoil(\gamma)))$

$\varphi_3 = goal(Have(Soil) \wedge Have(Rock)) \vee Have(\mathrm{Image}))$

⇩

$$\psi_1 = \psi_1 \lhd \psi_2 \lhd \psi_3$$

⇩

$\chi_1 = \psi_1 \wedge \neg \psi_2$ $\chi_2 = \psi_1 \lhd \psi_2 \lhd \psi_3$

Solving: Answer-Set Programming

[Tran & Pontelli, 2003]

Compilation using Answer-Set Programming:

- Set of **smodel** rules to encode and check if a basic desire **s** is satisfied by a trajectory **t**
- Using **maximize** constructs of smodel with an *admissible weighting* function **w** to compute the most preferred trajectory:

$$\alpha, \beta : \text{trajectories}; \psi : \text{general preference}$$

$$\alpha \prec_\psi \beta \rightarrow w_\psi(\alpha) > w_\psi(\beta)$$

$$\alpha \approx_\psi \beta \rightarrow w_\psi(\alpha) = w_\psi(\beta)$$

» Find plan trajectory a that maximize w(a)

Admissible Weighting Function

Basic desire φ : $\quad$ $if\ \ \alpha \mapsto \varphi \quad then \quad \mathrm{w}_{\varphi}(\alpha) = 1\ \ else\ 0$

Atomic preference $\quad \psi = \varphi_1 \lhd \varphi_2 \lhd ..... \lhd \varphi_n :\ w_{\psi}(\alpha) = \sum_{r=1}^{k} (2^{k-r}.w_{\varphi r}(\alpha))$

General preference :

$$\psi = \psi_1 \wedge \psi_2 \ \ or\ \ \psi_1 \vee \psi_2 \rightarrow w_{\psi} = w_{\psi 1} + w_{\psi 2}$$

$$\psi = !\psi_1 \rightarrow w_{\psi} = \max(\psi_1) - w_{\psi 1}$$

$$\psi = \psi_1 \lhd \psi_2 \rightarrow \max(\psi_2).w_{\psi 1} + w_{\psi 2}$$

Forward Heuristic Search

[Bienvenu et. al., 2006]

First-order preference language, extend PP language with:
quantification, variables, conditional construct, aggregators

- Use similar admissible weighting function (with degree of satisfaction) to guide best-first-search: **f** = **g** + **h**
  - **g** value: how well general preference has been satisfied with the current partial plan
  - **h** value: optimistically assuming all the unsatisfied parts of the general preference *will* be satisfied in the future:
    - Use *pessimistic* in addition to optimistic estimation to bound the plan quality, and to estimate weight for negation

SAT Compilation

[Giunchiglia & Maratea 2007]

control branching decisions in SAT formula **?** with horizon **n**
(like [Brafman & Chernyavsky, 2005] for CSP encoding)

Preference: any SAT formula (can be combination of trajectory and goal preference)
Preference orderings: similar to atomic preferences in [Tran & Pontelli, 2003]

Solving:

1. Create one Boolean variable **v(p)** for each preference **p**.
2. Variable ordering: if **p › p'** then select **v(p)** to branch before **v(p')**
3. Value ordering: **p = T** before **p = F**

» Like [Brafman & Chernyavsky] for binary CSP. Utilizing efficient SAT solvers.

Planning: Outline

- Background
- Qualitative Preferences
- Quantitative Preferences
  - OSP: Action cost vs. goal utility model
  - Trajectory preferences
    - Utility on facts visited by plan trajectory
    - PDDL3: cost on trajectory constraint violation
- Extensions

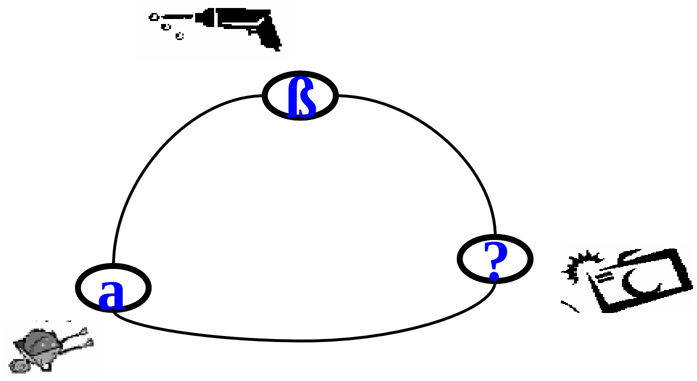
Qualitative vs. Quantitative Preferences

- **Qualitative**: appropriate when it is hard/undesirable for lay users to specify quantitative values of goals
  - Prefer red car to green car?
  - ~~How much "value" (in terms of money) of red or green car ?~~
    - » Lay user: Don't know.
    - » Car maker, market researcher: Do know

- **Quantitative**: applications where exact/approximate benefit for different decisions in terms of utility (money) are needed and quantitative preference values available.

Preference Model: Oversubscription Planning (aka Partial Satisfaction Planning)

[Smith, 2004; van den Briel et. al. 2004]



- *Soft*-goals with utilities:
  U(Have(Soil)) = 20, *U(Have(Rock))* = 50, *U(Have(Image))* = 30
- Actions with costs:
  C(Move(a,ß)) = 10, *C(Sample(Rock,ß))* = 20
- Objective function: find plan P that
  Maximize U(P) – C(P)

Find : (1) the best (most beneficial) subset **S** of goals
(2) the best (lowest cost) plan satisfying **S**

Modeling as MDP / MILP: Obvious ideas that don’t scale

[van den Briel et. al. 2004]

Goal utility/reward + action cost + optimization of utility-cost tradeoff
? **MDP** or Mixed Integer Programing

MDP

- No probability:
  - » Deterministic MDP
- Prevent recursively rewards collection:
  - » Define appropriate sink states

Property: global optimal
Performance: does not scale up well.

MILP

- Basic IP Encoding:

  Binary {0,1} variables for Actions/Facts

  Constraints:

  »**V(a)** = 1 ? **V(Pre(a))** = 1; **V(Effect(a))** = 1

  »**V(p)** = 1 ? **S V(a) = 1**; p in Effect(a)

  »**V(p)** = 1 : **p** is goal or in initial state

  Objective function: **minimize S V(a)**

- IP Encoding for OSP:

  –**V(p)** = {0,1} : **p** is goal

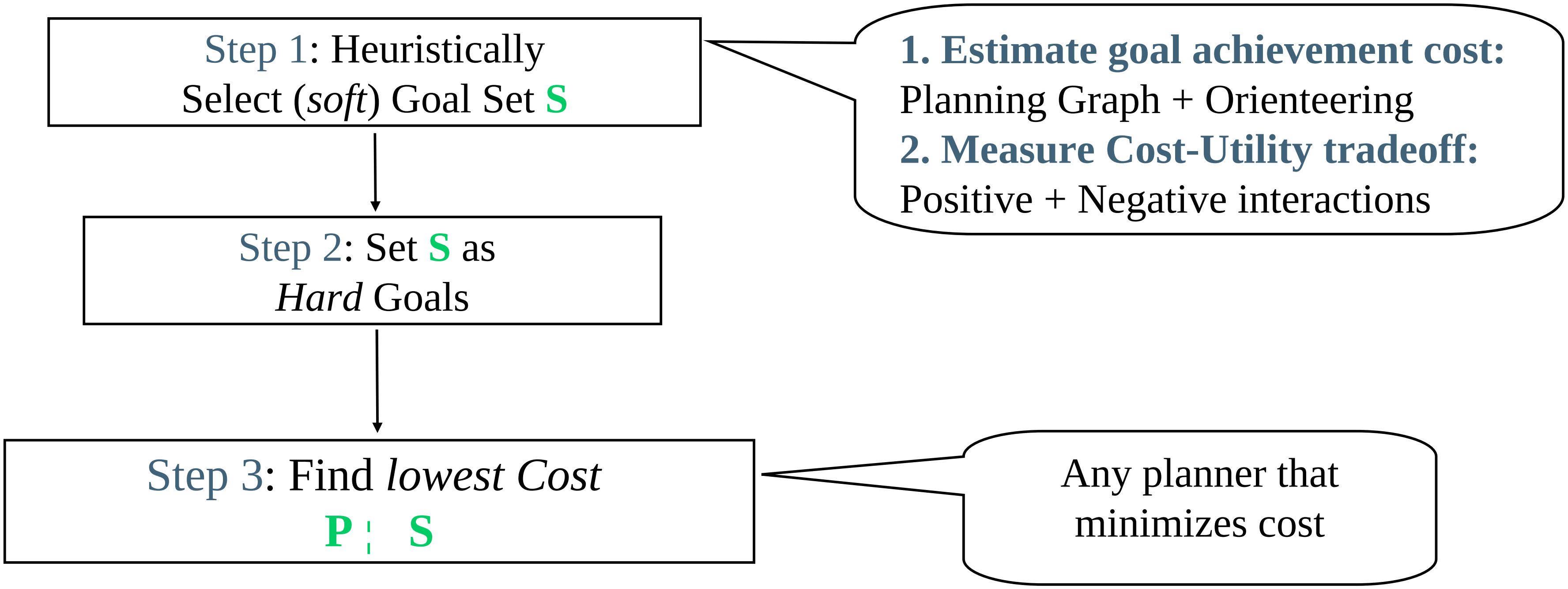
  –**maximize S V(g).U(g) - S V(a).C(a)**

Property: bounded horizon optimal
Performance: scale up better

Heuristic Search: Convert Soft Goals ? Hard Goals

[Smith, 2004; van den Briel, 2005; Sanchez, 2006]

MDP, MILP: supports OSP naturally, but do not scale well.
Heuristic search: scale well in non-OSP planning
? **Adaptation to OSP**: estimate best goal set *then use* non-OSP planner



Step 1: Heuristically Select (*soft*) Goal Set **S**

Step 2: Set **S** as *Hard* Goals

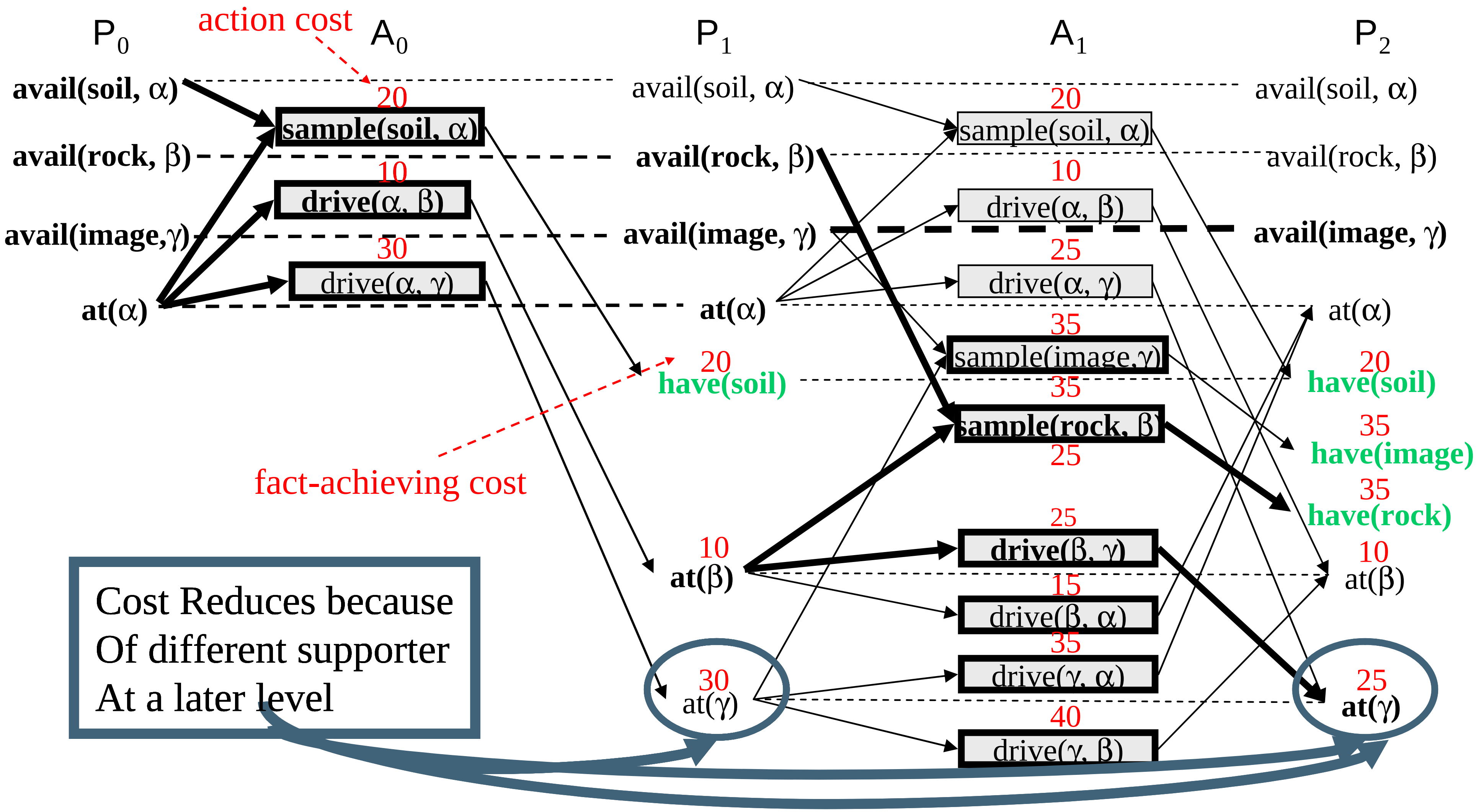
Step 3: Find *lowest Cost* **P ¦ S**

1. Estimate goal achievement cost:
Planning Graph + Orienteering
2. Measure Cost-Utility tradeoff:
Positive + Negative interactions

Any planner that minimizes cost

Goal Achievement Cost Estimate:
Cost Propagation on (relaxed) Planning Graph

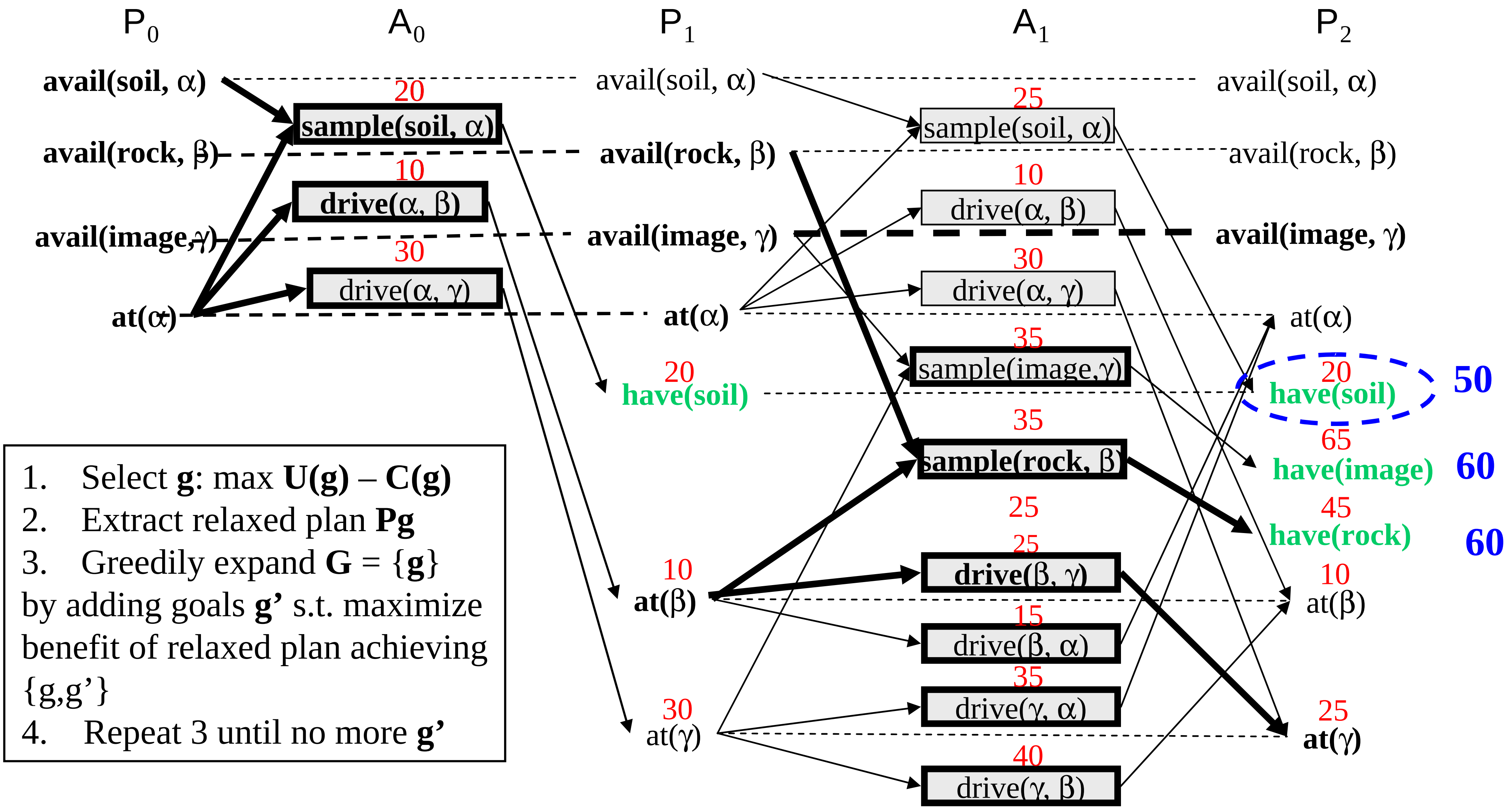
[Do & Kambhampati, 2002]



Cost Reduces because
Of different supporter
At a later level

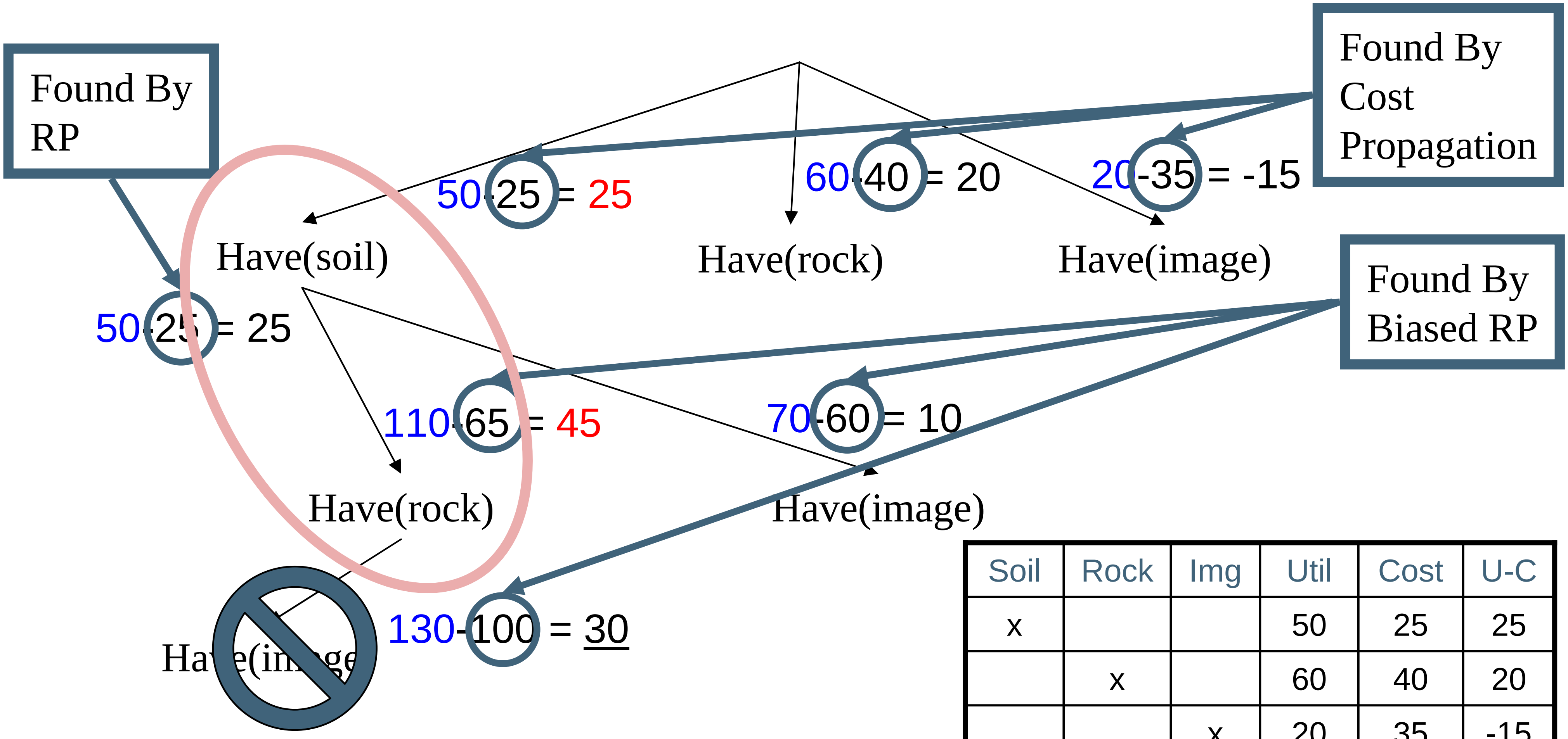
AltAlt: Goal Selection using Propagated Cost

[van den Briel et. al. 2004; Sanchez 2005]



1. Select **g**: max **U(g)** – **C(g)**
2. Extract relaxed plan **Pg**
3. Greedily expand **G** = {**g**} by adding goals **g'** s.t. maximize benefit of relaxed plan achieving {g,g'}
4. Repeat 3 until no more **g'**

AltAlt: Goal Set Selection



Soil	Rock	Img	Util	Cost	U-C
x			50	25	25
	x		60	40	20
		x	20	35	-15
x	**x**		**110**	**65**	**45**
x		x	70	60	10
x	x	x	130	100	30

AltWlt: Adjusting for Negative Interactions

[Sanchez 2005]

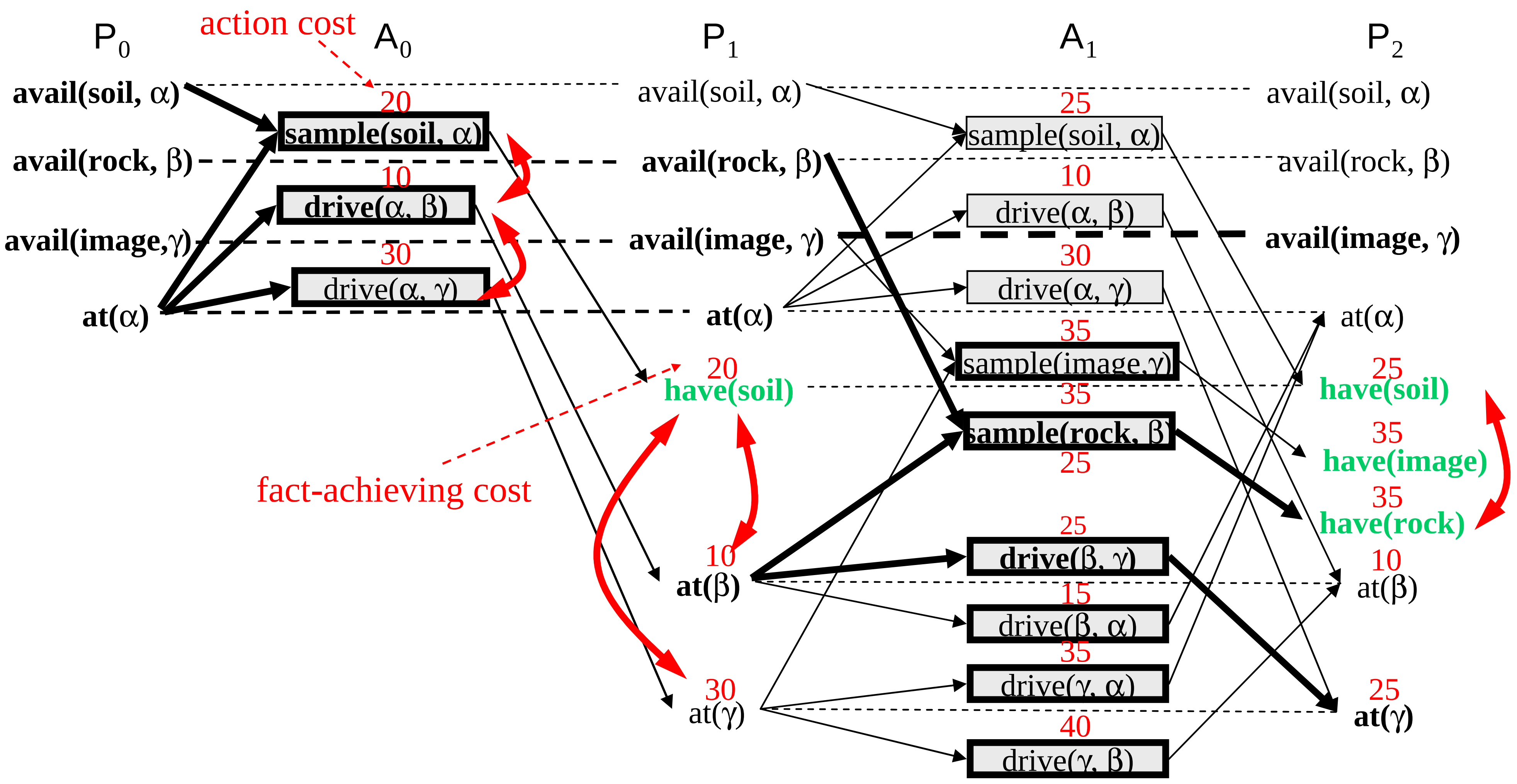
AltWlt: *Improve AltAlt^{PS} with Mutexes*

- Problem with AltAlt^{PS}:
  - What if the a priori goal set is not achievable because of negative interactions?
  - What if greedy algorithm gets bad local optimum?
- Solution: compare goal sets, instead of building one incrementally & greedily
  - Do not consider mutex goals
  - Add penalty for goals whose relaxed plan has mutexes.
    - » Use interaction factor to adjust cost:

      $\mathbf{Max}_{(g1,\ g2)}$ {lev(g1, g2) – max(lev(g1), lev(g2)) }
  - Find best goal set containing each goal

Mutexes in Planning Graph

[Do & Kambhampati, 2002]

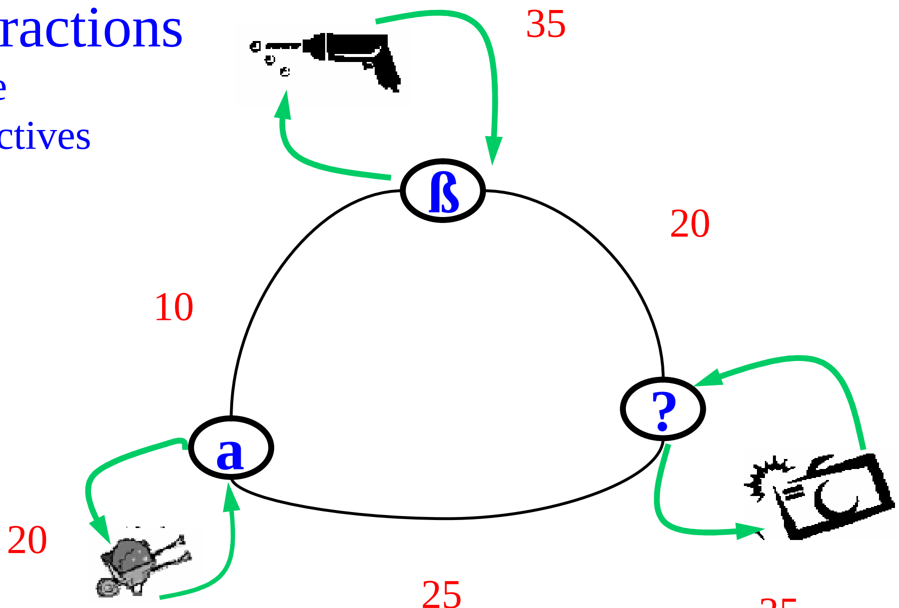


PG + Orienteering Problem

[Smith, 2004]

» Improve with more negative & positive interactions

- **Negative:** can not move to two locations at the same time
- **Positive:** move to one location can achieve multiple objectives



Orienteering Problem (variation of TSP):

- Set of linked cities
- Prize to visit each city
- Maximize reward with limited "gas"

» Suitable for "Transportation" domains

1. **Cost-propagation**: estimate cost to do experiment at each location
2. **OP**: use path-planning to build the orienteering graph
3. **Solve OP** and use the results to select goals and goal orderings

PG + OP: Generalization

[Smith, 2004]

- Abstraction: select subset **L** of *critical* literals
  - Based on relaxed plan analysis
- Build *state-transition graph* **G** based on **L** (project the state-space on **L**)
  - Set **G** as an orienteering graph
- Based on solving **OP** and relaxed plan at each node, select:
  1. Beneficial goal (sub)set **S**
  2. Order in which goals in **S** need to be achieved

➢ Planning search guided by goal ordering received from solving OP

Soft-Goal → Hard Goal: Summary

AltAlt^{PS}: Relaxed Plan
[van den Briel, 2004]

⇩

AltWlt: Relaxed Plan + Mutex Penalty
[Sanchez & Kambhampati, 2005]

⇩

OP + Relaxed Plan: more interactions through OP
[Smith, 2004]

» Better estimate of the most beneficial goal set (before actually find the plan)

Sapa^{PS}: Anytime BFS Forward Search

[van den Briel et. al. 2004; Do & Kambhampati 2004]

What if AltAlt's estimated goal set is bad?

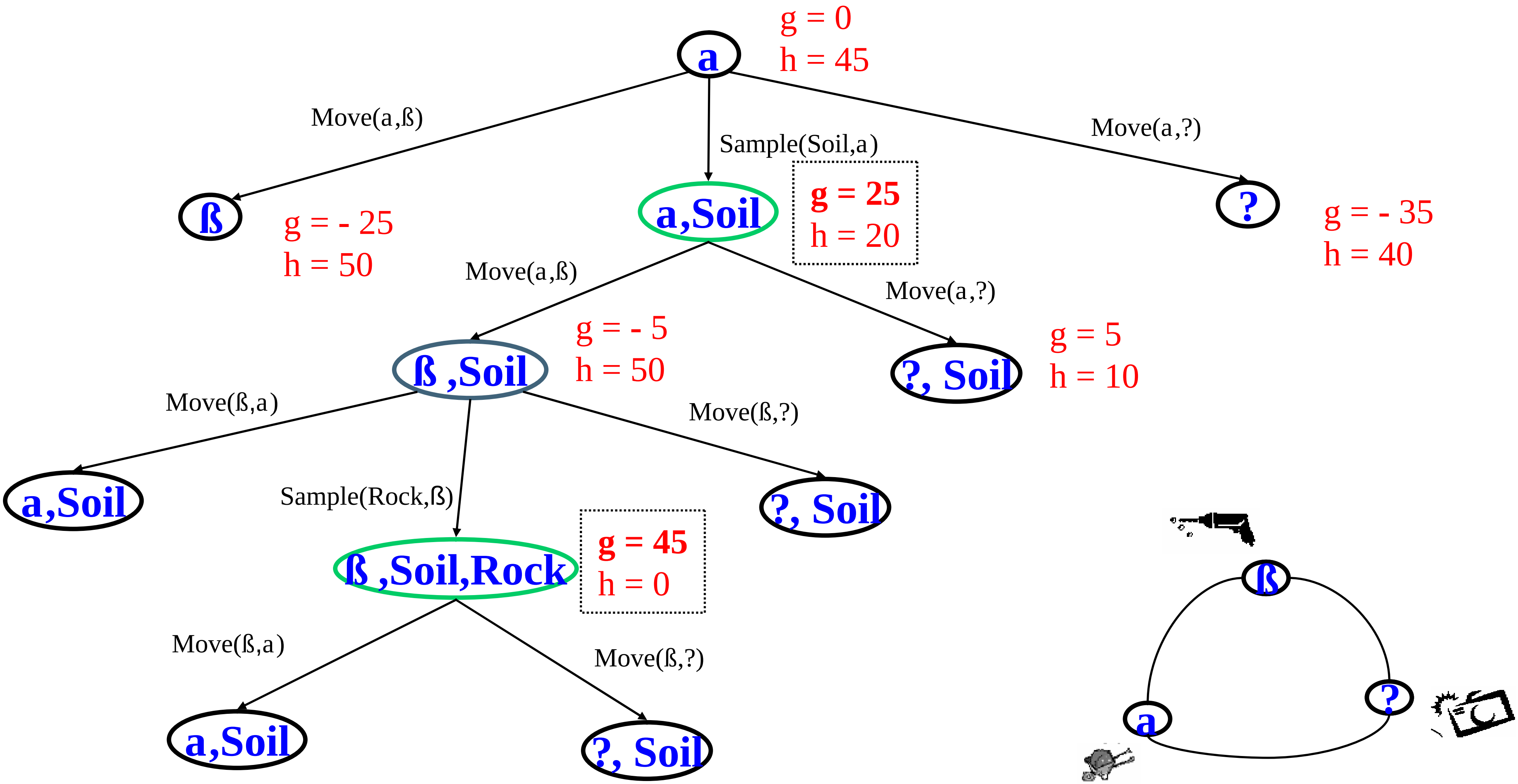
? BFS that can recover from bad initial estimation

Anytime: take advantage of a loose (soft) goal satisfying condition

- Search incrementally while looking for better solutions
  - Return better solutions as they are found (any node can be solution)
- Variation of **A***: **f = g + h** (with negative edge cost)
  - **Edge cost** (S,a,S'): (Util(S') – Util(S)) – Cost(a)
  - **g**-value: net-benefit of (real) plan so far
  - **h**-value: (relaxed plan) estimate of benefit to go to achieve the best goal set
    - » Relaxed plan found for all goals
    - » Iterative goal removal, until net benefit does not increase
  - **Anytime:** returns plans with increasing **g**-values.
  - If we reach a node with **h** = 0, then we know we can stop searching (no better solutions can be found)
    - » Optimal if **h** is admissible (over-estimate)

Sapa^{PS}: Example

[van den Briel et. al. 2004; Do & Kambhampati 2004]



Extending OSP: Trajectory Preference

[Bonet & Geffner, 2006]

Extension of Utility Model: At-end Goals ? Literals visited by plan trajectory

- Cost:

$$\forall a \in A : \text{cost}(a) > 0 : \text{penalty}$$

$$\forall p \in P : \text{cost}(p) : \text{penalty/reward}$$

- Plan Quality:

$$\min : \text{cost}(\pi) = \sum_{a \in \pi} \text{cost}(a) + \sum_{p \in F(\pi)} \text{cost}(p)$$

$F(\pi)$ facts made true *at some point* along the trajectory of plan π

Heuristic Search with Trajectory Preference

[Bonet & Geffner, 2006]

- Same search algorithm as **Sapa**^{PS}, but with different heuristic
  - **g** value: cost/reward of <u>all facts</u> **F** “visited” by the partial plan
  - **h** value: cost/reward of <u>all facts</u> **F’\F** visited by the relaxed plan
- Relaxation:
  - No conditional effect with negative cost:
    - » *Optimal relaxed plan* **h**$^+$ is *admissible*
      - ? A “most beneficial” plan when there is no delete effects.
  - With conditional effect: **h**$^+$ is *not* admissible
    - » Encoding the **h**$^+$ heuristic as rank of corresponding d-DNNF theory
      - *Stratified Encoding*: Horizon bounded
      - *LP encoding*: no horizon
    - » Theory rank can be computed in polynomial time according to the size of the encoding

PDDL 3: (Soft) Constraints on Trajectory

[Gerevini & Long, 2006]

- PDDL 3:
  - PDDL: Planning Domain Description Language (used in IPCs)
  - PDDL 3: Extended PDDL 2.2 with goal and trajectory preferences
- Trajectory Constraints:
  - Constraints over **entire trajectory** (instead of final goal state reached)
  - **Temporal Logic operators** on trajectory: always, sometimes, at-most-one, at-end, within, sometime-before, sometime-after, hold-during, hold-after
    - » Traditional planning goals: **at-end** conditions (default)
  - No preferences for action occurrence:
    - » Caveat: allows preferences over action’s preconditions
    - ? Can use dummy precondition to encode action cost in OSP
  - No nested operator
    - » Allowed in TLPlan & PP preference language
  - Syntax: **:constraints** ‹GD› (GD: goal description)
    - » Can be used separately, or can be merged with traditional **:goals** specification

PDDL 3: Preference = Soft Constraint

- Mixed hard/soft constraints:
  - Hard constraint on goal description: ‹GD›
  - Soft constraint: (preference [name] ‹GD›)
- Applicability:
  - Goals (at-end conditions)
  - Constraints on trajectory
  - Action preconditions preferences
- Preference & Plan Quality:

Preference name can be reused/shared

$$(\textit{preference}\ \text{HaveSoil}\ \textit{Have}(\textit{Soil}))$$

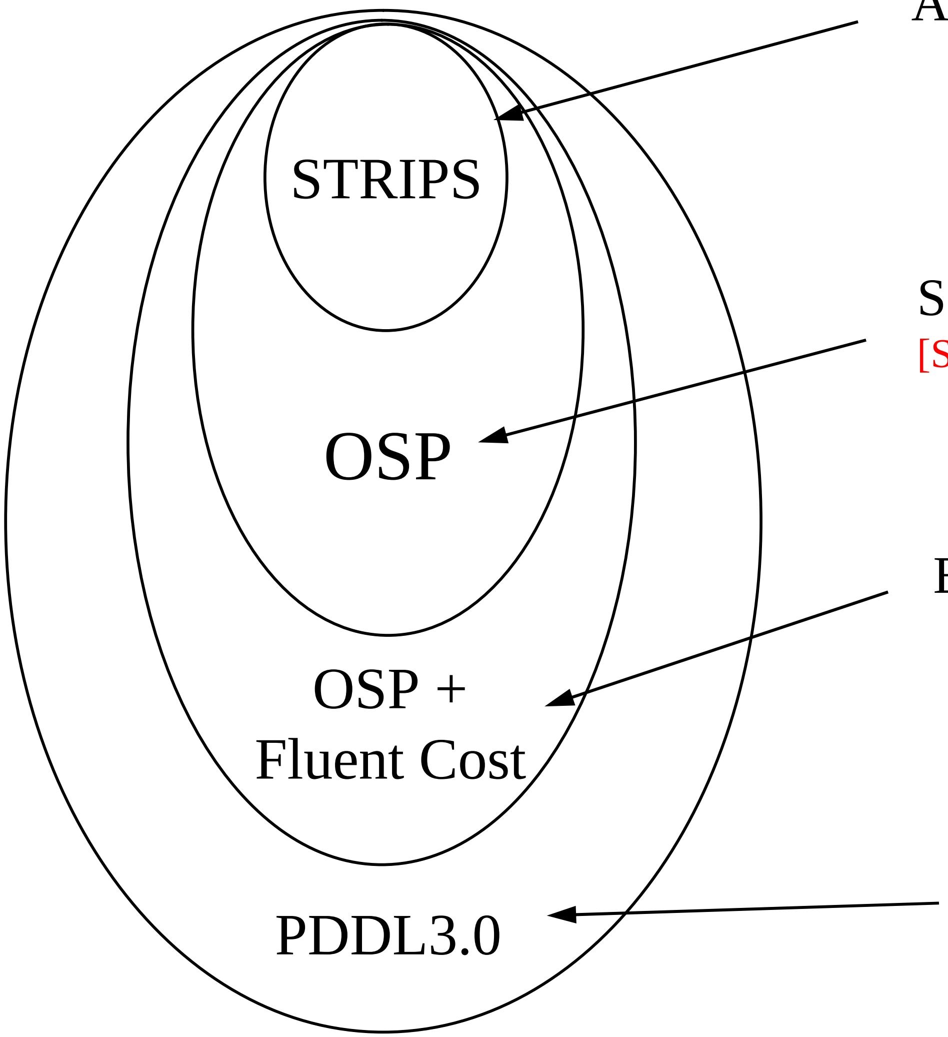
$$(\forall\,(?R\text{ - }\textit{rover})\,(\textit{preference}\ \textit{Visit}\alpha\ (\textit{sometime}\ \ (\textit{at}\ ?R\ \alpha))))$$

$$\begin{aligned}:\textit{metric}\ \text{mininize}\ (+\,(\times 10\ \ \textit{makespan}) &+ \\ (\times 5\,(\textit{is-violated}\ \ \text{HaveSoil})) &+ \\ (\times 0.5\,(\textit{is-violated}\ \ \textit{Visit}\alpha)))&\end{aligned}$$

number of time this pref is violated

may involve more than one preferences

Modeling Quantitative Preferences: Evolution



STRIPS

OSP

OSP +
Fluent Cost

PDDL3.0

All hard goals

Soft goals with utility + action cost
[Smith, 2004; van den Briel et. al. 2004]

Extend goal utility model:
at-end ? **some-times**
[Bonet & Geffner, 2006]

most of temporal logic on
plan trajectory + implicit action cost
[Gerevini & Long, 2006]
(quantitative version of qualitative model PP)

PDDL3: Examples

Mixed hard/soft goals:

hard constraint

soft constraint

$$(:goal\ \ (and\ Have(Soil))\ \ (preference\ \ \mathrm{Pref}_1\ \ Have(\mathrm{Image}))))$$

Action precondition preferences:

$$(:\mathrm{condition}\ \ (and\ \ at(Rover,\alpha)\ \ (preference\ \ \mathrm{Pref}_3\ \ Avail(Soil,\alpha)))$$

Mixed hard/soft trajectory constraints:

$$\begin{aligned}(:\mathrm{constraint}\ \ (and\ \ &(sometime\text{-}after\ \ at(R,\alpha)\ \ at(R,\beta))\\ &(preference\ \ \mathrm{Pref}_2\ \ (within\ \ 2\ \ at(R,\gamma))))\end{aligned}$$

Plan Quality: satisfaction of hard trajectory constraints
+ minimization of accumulated cost of preference/soft constraints violations

$$\begin{aligned}:metric\ \mathrm{mininize}\ (+\ (\times 10\ total-time)\ +\\ (\times 5\ (is\text{-}violated\ \ \mathrm{Pref}_1))\ +\\ (\times 0.5\ (is\text{-}violated\ \ \mathrm{Pref}_2)))\end{aligned}$$

PDDL3.0 Preference: Solution Approaches

- Simplification/Conversion:
  - Preference violation cost ? OSP goal utility
  - Action condition preferences ? goal preferences
  - Trajectory preferences ? simple preferences
- Compilation:
  - Extend PDDL2 Encodings: SAT, ILP
- Heuristic search:
  - Extend plan validation**:** check for **hard constraints** on trajectory
  - Adjust heuristic estimate: account for **preference violation cost**

Yochan^{PS}: Simple Preferences ? OSP

[Benton et. al. 2006]

- Goal preference **g** with “violation” cost **C(g):**
  - » New action **Pref$_g$**: **g ? g’** with **Util(g’) = C(g)**

    ? do not achieve g’ loose utility amount C(g)
- Action condition preference:
  - » **a** ? **S(a)**: set of action with no condition preference by removing a subset of preferred conditions and make other hard constraints

    Example: $a_{\{p,q\}} \Rightarrow S(a) = \{a^0_{\{\}}, a^1_{\{p\}}, a^2_{\{q\}}, a^3_{\{p,q\}}\}$
  - » If a subset **S’** of **S(a)** applicable in a given state, only apply the *least cost* action

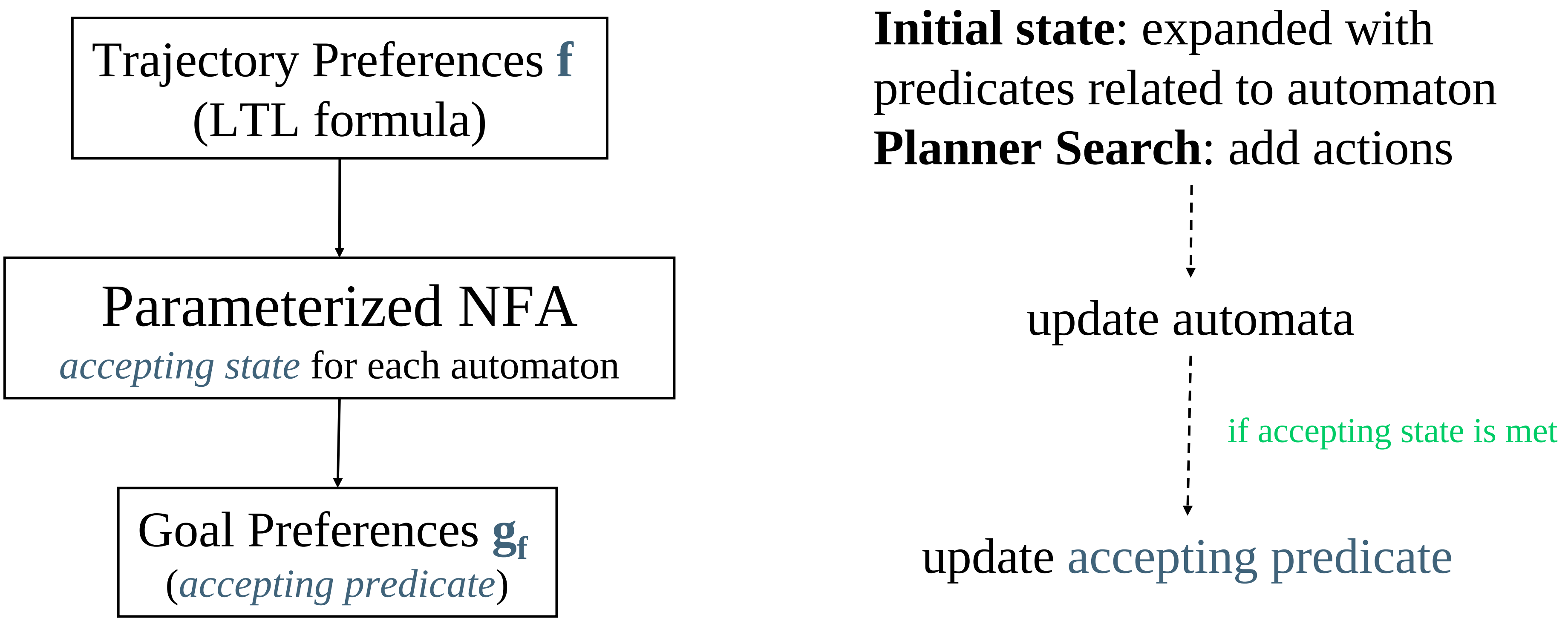
    Example: $\{p, r\} \Rightarrow a^0_{\{\}}, a^1_{\{p\}}$

    least cost among these 2 applicable actions

Use Sapa^{PS} to solve the simplified problem

LTL with Preferences ? Simple Preferences

[Baier, Bacchus, & McIlrath, 2007]



Trajectory Preferences **f** (LTL formula)

Parameterized NFA
accepting state for each automaton

Goal Preferences **g_f** (*accepting predicate*)

Initial state: expanded with predicates related to automaton
Planner Search: add actions

update automata

if accepting state is met

update accepting predicate

Action precondition preferences: count the number of time violated when applying actions
Goal (at-end) preferences: add new **accepting predicate** (no automaton)
PDDL3.0 Objective: refer to **accepting predicates** and **counter**

(similar systems: MIPS-BDD, MIPS-XXL [Edelkamp et. al., 2006])

SAT: Quantitative → Qualitative

[Giunchiglia & Maratea 2007]

Quantitative Preference: convert **quantitative ?   qualitative** preferences using SAT variable representing possible objective function values

P : Set of preferences

$c(p)_{p \in P}$: utility of satisfying a preference $p \in P$

Quality of plan $\prod$: $\sum_{\prod \mapsto p} c(p)$

Maximum achievable plan quality : $\mathrm{C} = \sum_{p \in P} c(p)$

Possible C value can be reprensented by $\mathrm{n} = \lceil \log_2(C) + 1 \rceil$ bits $\mathrm{b_n \ldots b_1}$

(Qualitative) preferences : $\mathrm{b_n \succ b_{n\text{-}1} \succ \ldots.. \succ b_1}$

Finding optimal plan according to the qualitative preferences by $\mathrm{b_i}$

ILP: Simple Preferences

[van den Briel et. al., 2006]

- Logical expression
  - A
  - $\neg A$
  - $A \vee B$
  - $A \rightarrow B \quad \leftrightarrow \quad \neg A \vee B$
  - $A \wedge B$

- LP expression
  - $A \geq 1$
  - $(1 - A) \geq 1$
  - $A + B \geq 1$
  - $A \leq B \quad \leftrightarrow \quad (1 - A) + B \geq 1$
  - $A \geq 1,\, B \geq 1$

1. Obtain logical expression of the preference
2. Transform expression into CNF (SAT)
3. Formulate ILP constraints corresponding to SAT clauses
4. Set up the objective function based on preference violation cost

(lot of examples in [van den Briel et. al. 2006])

Tutorial Outline

- Introduction (15)

- Oversubscription in Scheduling: Basic view (Terry) (75m)

(Break)

- Oversubscription Planning: Basic view (Minh)(75min)

- ➡ Extensions for Scheduling: Reasoning over quality metrics (Terry) (20m)

- Extensions for Planning: Metric goals, goal dependencies etc. (Minh)(20m)

Scheduling

Planning: Outline

- Background
- Qualitative Preferences
- Quantitative Preferences
- Extensions
  - Goal utility dependencies
  - Metric planning: degree of satisfaction
  - Temporal planning: soft deadlines

Non-Independent Goals

[Do et. al., 2007]

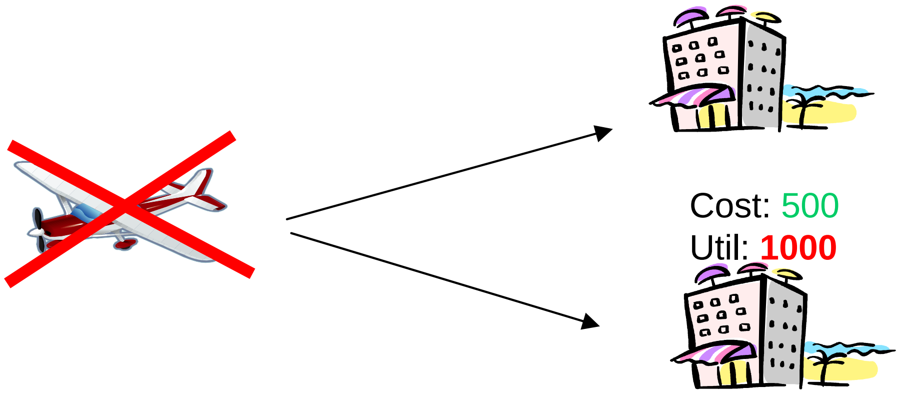
Cost: 10
Util: **20**

Cost: 100
Util: **50**

Cost: 110
Util: **300**

Cost: 15000
Util: **30000**

Cost: 45000
Util: **35000**



Cost: 500
Util: **1000**

Cost: 500
Util: **0**

Modeling Goal Dependencies

General Additive Independence (GAI) Model [Bacchus & Grove, 1995]

Utility over sets of dependent goals

$$S \subseteq G \longrightarrow f(S) \in R$$

$$U(G) = \sum_{S \subseteq G} f(S)$$

Util: 20

Util: 50

Util: 300

$f(So) = 20$ $f(Sh) = 50$ $f(\{So, Sh\}) = 230$

$$U(\{So, Sh\}) = 20 + 50 + 230 = 300$$

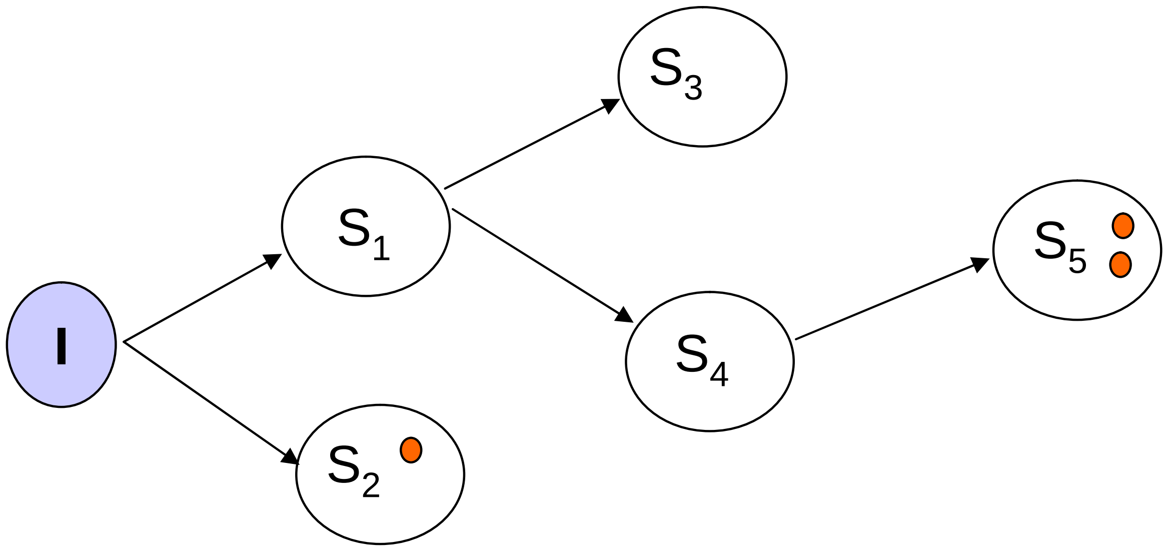
iPUD : Mixed Integer Linear Programming Encoding

[Do et. al., 2007]

- Remove hard constraints on goal achievement.
- Introduce a new binary variable for each related goal set *S*.
- Add constraints to ensure that *S* is achieved when $\forall g \in S$ achieved (and vice versa).
- New *objective function* capturing goal utility dependencies.

SPUDS: Forward Heuristic Search

Extending Sapa^{PS} (2004)



Forward State-space Planning Search using **A***

Node evaluation: *$g = U(G_S) – Cost(P_S)$* | *h(S): expected additional benefit*

Output better quality solutions given more time (anytime)

SPUDS: Heuristic

[Do et. al., 2007]

- Approximate the relaxed plan with the best utility-cost tradeoff

Step 1: Estimate the *lowest cost relaxed plan* P^+ achieving **all** remaining goals

⇩

Step 2: Build *cost-dependencies* between goals in P^+

⇩

Step 3: Find the *optimal relaxed plan* within P^+
Using ILP encoding of P^+

Admissible Heuristic: Optimal Relaxed Plan by ILP Encoding

[Benton et. al., 2007]

- Planning Relaxation:
  - All action preconditions and effects are not relaxed
  - Action orderings are relaxed
- Heuristic: **LP relaxation** + **Planning Graph**
  1. Relax the integer constraints on variables:
     - Objective function value as admissible bound to prune nodes
  2. Pick action set **S** and goals with values returned by LP relaxation beyond certain threshold
  3. Extract the relaxed plan using the relaxed planning graph biased by goals and actions returned in Step 2.

⇨ finding good quality solutions faster; better pruning bounds

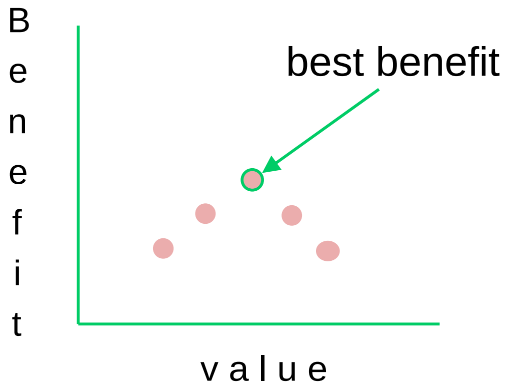
Metric Goals: Degree of Satisfaction

[Benton et. al., 2005]

- Numeric goals also have utility:
  - More soil gives better instrument reading
    - » $\mathbf{U(Have(Soil,}a\mathbf{)) = f_u(Amount(Soil,}\ a\mathbf{))}$
- Cost for achieving varying values differs
  - More images take more memory and communication bandwidth
    - » $C(TakeImage(a)) = f_c(Size(Image(a)))$

Objective

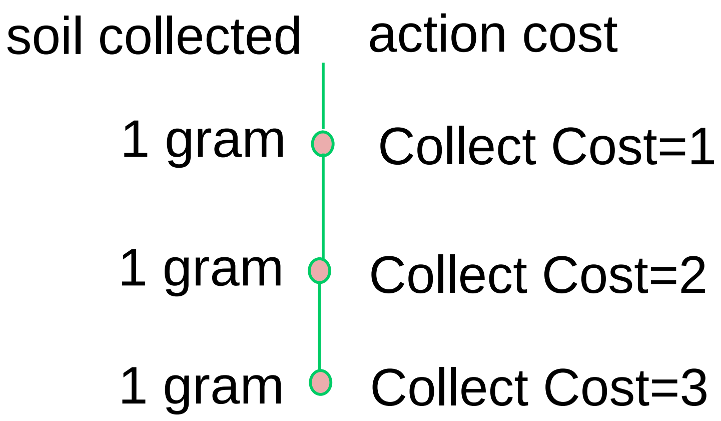
Satisfy numeric goals at different values to give varying utility



- Want more/less

  G = soil-sample ? [2,4]

  U(G) = 2 * (soil-sample)
- **Challenge:** A measurable level of numeric goal achievement: *degree of satisfaction*



soil collected	action cost
1 gram	Collect Cost=1
1 gram	Collect Cost=2
1 gram	Collect Cost=3

Soil = 2 gram: B = 2*2 – (1+2) = 1

Soil = 3 gram: B = 2*3 – (1+2+3) = 0

PG-based Heuristic

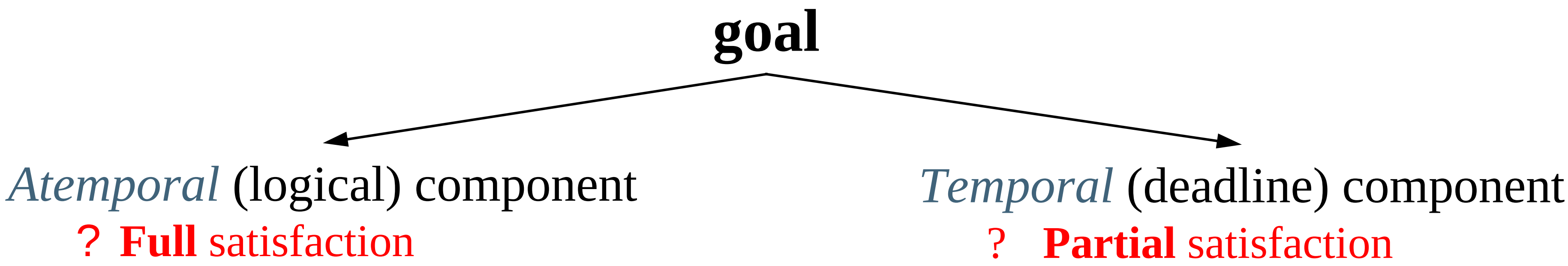
[Benton et. al., 2005]

- Metric temporal planning graph
- For each metric fluent variable:
  - Cost-propagation to estimate cost to achieve each reachable fluent value, bound cost depends on
    - » Cost to achieve lower value
    - » Action cost leading from lower value
- Extract the relaxed plan:
  - Start with best benefit bounds
  - Relaxed plan includes
    - » Actions
    - » Supporting bounds

Temporal Planning: Preference on Goal Achievement Time

[Hanks & Williamson, 1994]

PSTC: Partial Satisfaction of Temporal Component



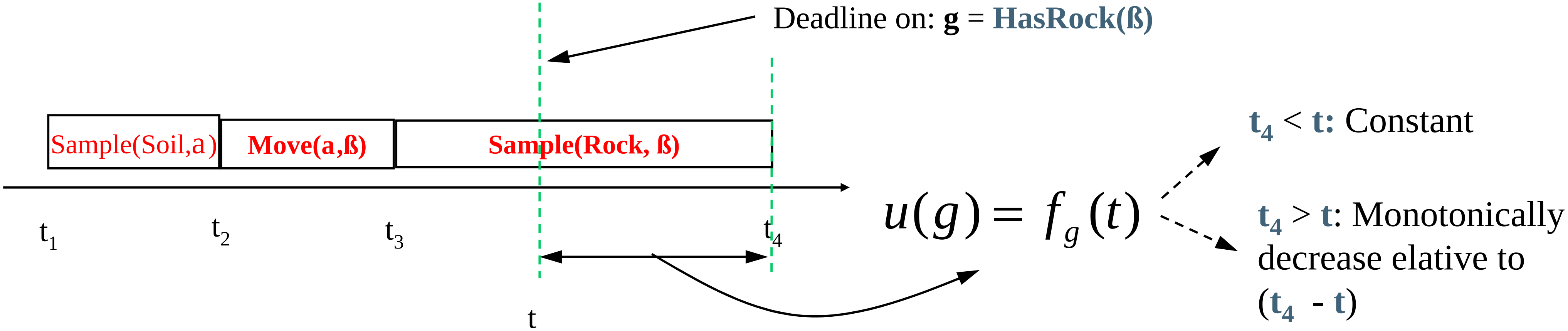
goal

Atemporal (logical) component

? **Full** satisfaction

Temporal (deadline) component

? **Partial** satisfaction



Deadline on: **g** = **HasRock(ß)**

Sample(Soil,a)

Move(a,ß)

Sample(Rock, ß)

t_1 t_2 t_3 t_4

t

$$u(g) = f_g(t)$$

$\mathbf{t_4} < \mathbf{t}$**:** Constant

$\mathbf{t_4} > \mathbf{t}$: Monotonically decrease elative to ($\mathbf{t_4}$ - $\mathbf{t}$)

PYRRHUS Planner

[Hanks & Williamson, 1994]

- Extending UCPOP to handle PSTC problem (with numeric resources – action costs)
- Actions always consume time or resource
  - ? decrease total utility
  - ? Admissible Heuristic:
    - » Lower bound on the estimated total resource and time (makespan) consumption of remaining plan
- Search framework: Anytime BFS like *Sapa*^{PS}, but based on POP instead of forward state-space
- Pruning:
  - – Using "best plan so far" to prune open queue
  - – domain-dependent rules in UCPOP

References

- **Simulation-Based Planning for Planetary Rover Experiments**, D. Joslin, J. Frank, A. Jónsson, D.Smith. Proc. 2005 Winter Simulation Conf., 2005. *(Oversubscription planning by searching in the space of plan strategies)*
- **Choosing Objectives in Over-Subscription Planning**, D. Smith. Proc. 14th Intl. Conf. on Automated Planning & Scheduling (ICAPS-04), 2004. *(Describes a method for generating and solving an Orienteering Problem to choose the subset of goals for an oversubscription problem)*
- **The Next Challenges for AI Planning**, D. Smith. Lecture given at the Planet International Summer School, Madonna di Campiglio, Italy, June 2003. *(Presentation of six technical challenges for AI planning, motivated by NASA planning problems – one challenge is over-subscription planning)*
- **Over-subscription in Planning: A Partial Satisfaction Problem.** Menkes van den Briel, Romeo Sanchez Nigenda and Subbarao Kambhampati. ICAPS 2004 Workshop on Integrating Planning into Scheduling. *(describe two different approaches to solve over-subscription problem)*
- **Effective approaches for Partial Satisfation (over-subscription) Planning**. Menkes van den Briel, Romeo Sanchez Nigenda, Minh B. Do and Subbarao Kambhampati. AAAI 2004. *(describe three different approaches to solve over-subscription problem)*
- **Partial Satisfaction (Over-Subscription) Planning as Heuristic Search**. Minh B. Do and Subbarao Kambhampat. Proc. of Knowledge Based Computer Systems (KBCS), 2004. *(describe SapaPS in more details than the AAAI04 paper)*
- **Planning Graph Heuristics for Selecting Objectives in Over-subscription Planning Problems.** Romeo Sanchez and Subbarao Kambhampati. ICAPS 2005. *(more advanced heuristic taking negative interaction into account to improve AltAltPS)*
- **Oversubscription planning with metric goals**. J. Benton, Minh B. Do and S. Kambhampati. IJCAI 2005. (*extend SapaPS to deal with metric goals*)
- **Planning with Goal Utility Dependencies**. Minh Do, J. Benton, Subbarao Kambhampati, Menkes van den Briel. IJCAI 2007. *(with goal utility dependencies represented using GAI framework)*
- **Planning as Satisfiability with Preferences.** E. Giunchiglia, M. Maratea. In AAAI 2007.
- **Planning Graph based Reachability Analysis**. D. Bryce & S. Kambhampati. Tutorial given at ICAPS06.
- **Planning with Preferences and Trajectory Constraints by Integer Programming**. Menkes van den Briel, Subbarao Kambhampati, and Thomas Vossen. Workshop and Preferences & Soft Constraints. ICAPS 2006.
- **A Hybrid Linear Programing and Relaxed Plan Heuristic for Partial Satisfaction Planning Problems**. J. Benton, Menkes van den Briel, Subbarao Kambhampati. ICAPS07.

- **Stochastic Over-Subscription Planning using Hierarchices of MDPs**, N. Meuleau , and R. I. Brafman, and E. Benezara. In Proc. of ICAPS'06.
- **Planning with Goal Preferences and Constraints**, Ronen I. Brafman and Yuri Chernyavsky. In Proc. of ICAPS'05. (*CSP planning with preferences*)
- **A Heuristic Search Approach to Planning with Temporally Extended Preferences**, Baier, J. and Bacchus, F. and McIlraith, S., 2007. IJCAI07. (*compile away TL goal preferences then use heuristic search*)
- **Planning with Qualitative Temporal Preferences**, Bienvenu, M. and Fritz, C. and McIlraith, S., 2006. In Proc. of KR06.
- **Planning with First-order Temporally Extended Goals**. J. Baier & S. McIlraith. AAAI 2006.
- **CPP: A Constraint Logic Programming Based Planner with Preferences**. Phan Huy Tu, Tran Cao Son, and Enrico Pontelli. LPNMR 2007. LNAI 4483, pp. 290–296, 2007.
- **Planning with Preferences using Logic Programming**. Tran Cao Son and Enrico Pontelli. TPLP, Vol 6, Issue 5, September 2006, 559-608. (*qualitative preferences over trajectory of plan*)
- **Plan Constraints and Preferences in PDDL3: The language of the Fifth International Planning Competition**. Alfonso Gerevini & Derek Long. Technical Report, Univ. of Brescia, Italy, 2005.
- **Constraint Partitioning for Solving Planning Problems with Trajectory Constraints and Goal Preferences** C. Hsu, B. Wah, R. Huang, and Y. Chen**.** *Proc. IJCAI-07*, 2007.
- **YochanPS: PDDL3 Simple Preferences as Partial Satisfaction Planning**. J. Benton, Subbarao Kambhampati and Minh B. Do. IPC-5 Booklet.
- **Large-Scale Optimal PDDL3 Planning with MIPS-XXL**. Stefan Edelkamp, Shahid Jabbar and Mohammed Nazih. IPC-5 Booklet.
- **Optimal Symbolic PDDL3 Planning with MIPS-BDD**. Stefan Edelkamp. IPC-5 Booklet.
- **Planning with Temporally Extended Preferences by Heuristic Search**. Jorge Baier, Fahiem Bacchus and Sheila McIllraith. IPC-5 Booklet.
- **Optimal Planning with a Goal-Directed Utility Model**. Hanks & Williamson. In Proc. Of ICAPS-94.
- **Utility Models for Goal-Directed Decision-Theoretic Planners**. Haddawy & Hanks. Technical Report. CS Dept. Univ of Washington, 1993.
- **Preference Representation and Constrained Optimization with CP-Nets.** Ronen Brafman & Carmel Domshlak. *CP'2003 Tutorial.*
- **Heuristics for Planning with Penalties and Rewards using Compiled Knowledge**. B. Bonet and H. Geffner. Proc. of KR-06.
- **5th International Planning Competition: Results of the Deterministic Track**. Alfonso Gerevini, ICAPS 2006.

Planning & Scheduling with Over-Subscribed Resources, Preferences, and Soft-constraints

Minh Do, Terry Zimmerman, Subbarao Kambhampati

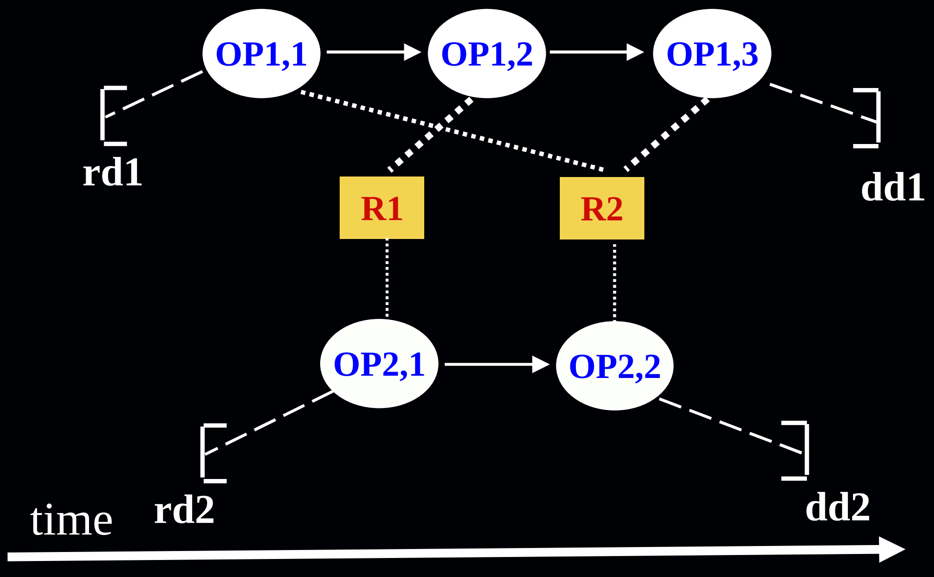
parc Palo Alto Research Center

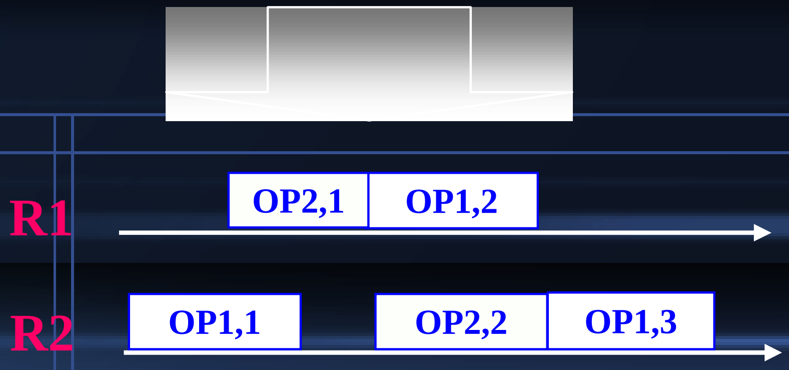
Carnegie Mellon THE ROBOTICS INSTITUTE

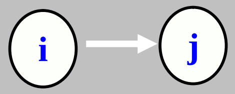
ASU

What is Scheduling?

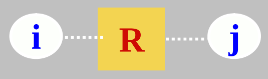
- Given problem constraints and objective criterion, figure out how to best execute tasks with available resources over time







$st(i) + p(i) = st(j)$, where $p(i)$ is the processing time of op i

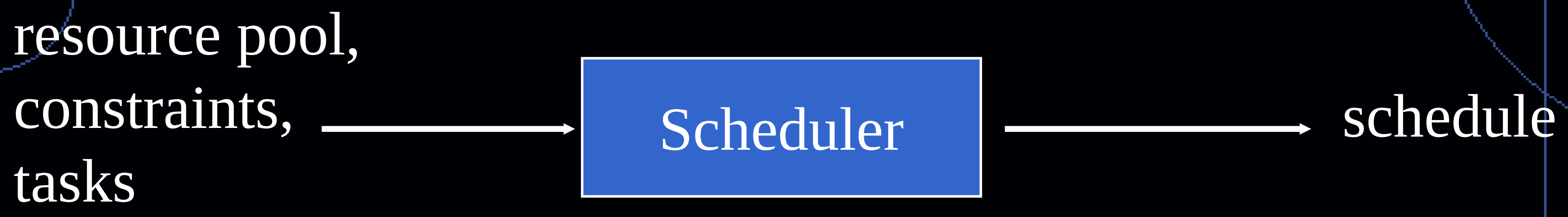


$st(i) + p(i) = st(j) \vee st(j) + p(j) = st(i)$

$rd(j) = st(i)$ for each op i of job j

Minimize ? $|c(j) - dd(j)|$

Scheduling



resource pool,
constraints,
tasks

Scheduler

schedule

User's point of view: Allocate as many tasks as possible, satisfying deadlines, priorities, maximizing a metric (at the lowest cost)!

Scheduler's point of view:

Not enough resources (e.g. time, aircraft) to allocate all tasks:

→ **"Over-subscribed"**

‘Classical’ Static Optimization Scheduling and Beyond…

For problems of *appropriate size* and flavor general CP-based solution approaches have been effective (focus of OR):

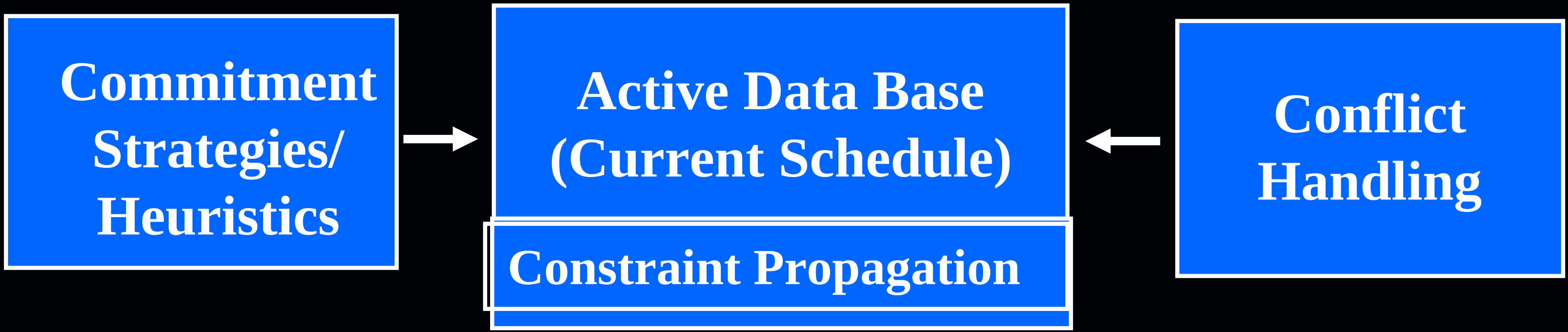
- Optimization is a primary interest
- Building the constraint model is a primary task
- Powerful general CP solvers then applied

Our focus: Practical scheduling problems that generally cannot be formulated as static optimization:

- Scale often precludes generation of optimal solutions
- Situated in a larger problem-solving context
- Dynamic, unpredictable environment
- Ongoing iterative process
- Multiple inter-dependent agents

Core Technologies for Practical Scheduling: The Last 10 Years

Constraint-based search procedures:



Commitment Strategies/ Heuristics

Active Data Base (Current Schedule)

Constraint Propagation

Conflict Handling

Key Capabilities:

- Dynamic, *incremental* solution change & (re)optimization
- Flexible, *user-directable* procedures
- *Rich representations* of operational constraints
- *Collaborative coordination* of multiple scheduling agents

Scheduling Issues

- Over-subscribed problems
- Preferences & Soft Constraints
  - ‘soft’ deadlines
  - Priority
  - Quality / Utility
  - Schedule stability
  - Schedule robustness
  - Learning preferences
  - Task-to-Task facilitation, hindering

Contention: Most of these issues have been inherent to practical scheduling problems for years…

Oversubscribed Problems

There are typically too many things to do and not enough time and resources to do them.

- Oversubscription may be localized to particular resources or phases of the schedule.
- New tasks can overwhelm previously undersubscribed resources.
- In a dynamic domain it may not be possible to easily determine the extent of oversubscription.

Over-subscribed Scheduling

A Long history of applications…

- NASA's shuttle ground processing (Deale et al., 1994)
- Telescope observations scheduling (Bresina, 1996), Hubble space telescope observations (Johnston, M., and Miller, G.)1994.
- Satellite observation scheduling (Frank, Jonsson, Morris, & Smith, 2001; Globus, Crawford, Lohn, & Pryor, 2003)
- USAF Air Mobility Command (AMC) airlift (Kramer & Smith, 2003)
- AFSCN -multiple resource satellite communications (Barbulescu, Howe, Roberts 2006)
- *Not to mention the OR oriented work…*

Soft constraints & preferences in scheduling

- Priority: User-specified preference over tasks
- Quality: Preference for high quality *(compare to priority)*
- Stability: preference for minimizing change in extant schedules
  - Humans are often part of the loop, executing the schedule and reviewing it.
  - Excessive scheduling and re-scheduling can be wasteful of resources.
- Robustness: preference for schedules that can absorb some degree of unexpected temporal variation without major rescheduling
- Facilitation, hindering interaction between tasks

Preferences & Soft Constraints in Scheduling

A Long history of applications…

- NASA's shuttle ground processing (Deale et al., 1994)
- Telescope observations scheduling (Bresina, 1996), Hubble space telescope observations (Johnston, M., and Miller, G.)1994.
- Satellite observation scheduling (Frank, Jonsson, Morris, & Smith, 2001; Globus, Crawford, Lohn, & Pryor, 2003)
- USAF Air Mobility Command (AMC) airlift (Kramer & Smith, 2003)
- AFSCN -multiple resource satellite communications (Barbulescu, Howe, Roberts 2006)
- *Not to mention the OR oriented work…*

Over-subscribed scheduling: Solution Approaches

Two classes of solution techniques have emerged:

I. Search directly in the space of possible schedules (*iterative repair*). Initial base solution is progressively revised and hopefully improved over time.

Examples: Iterative Repair, Task Swap

II. Search in space of task permutations:

a. Permutation specifies a scheduling *order*

b. Ordered tasks are transformed into an actual schedule by a “schedule builder” (which handles all domain constraints)

Examples: Squeaky Wheel Optimization, Genetic Algorithms,

Both permutation-space and schedule-space approaches have been shown to perform effectively in specific problem domains.

Over-subscribed scheduling (with preferences)

Consider two real-world problems…

- Airlift Allocation (AMC)

  Day-to-day allocation of aircraft and aircrews at the Air Mobility Command: Tasks (missions) have priorities, execution time windows, pickup and drop-off locations, task-specific multi-capacity resources and resource-specific set-ups.

- Air Force Satellite Control Network (AFSCN) Access Scheduling

  Input communication requests for Earth orbiting satellites must be scheduled on a total of 16 antennas spread across 9 groundbased tracking stations. No explicit notion of priority, all tasks are weighted equally.

Basic (AMC) Airlift Allocation Problem

Requests:

Mission1:

•pick up cargo at A

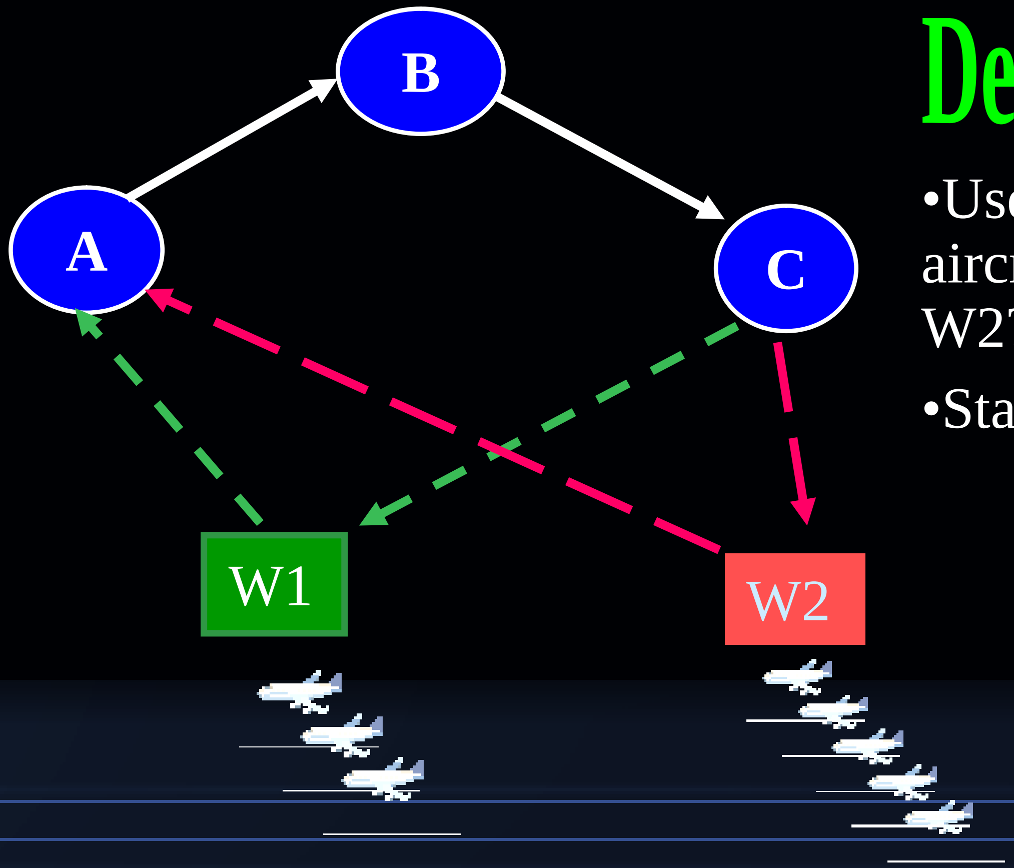
•deliver to B,

• then C.

Mission2

…

Mission-n



Decisions:

•Use resources (e.g., aircraft) from wing W1 or W2?

•Start at what time?

(Missions considered in strict priority order)

Over-subscribed scheduling (with preferences)

Domain Characteristics: Airlift Allocator

- Schedule is often over-subscribed (more missions than available capacity)
- No mission of lower priority can be in final schedule if an unassigned higher priority mission can be allocated. *So priority has both a 'hard'* and *'soft' constraint aspect..*
- Greedy search handles priority well, but can leave many "gaps" in the schedule.
- Schedule must be generated quickly and should be open to updates for:
  - New missions
  - Changes in the "state of the world"

Dealing with unassignable tasks in the face of over-subscription

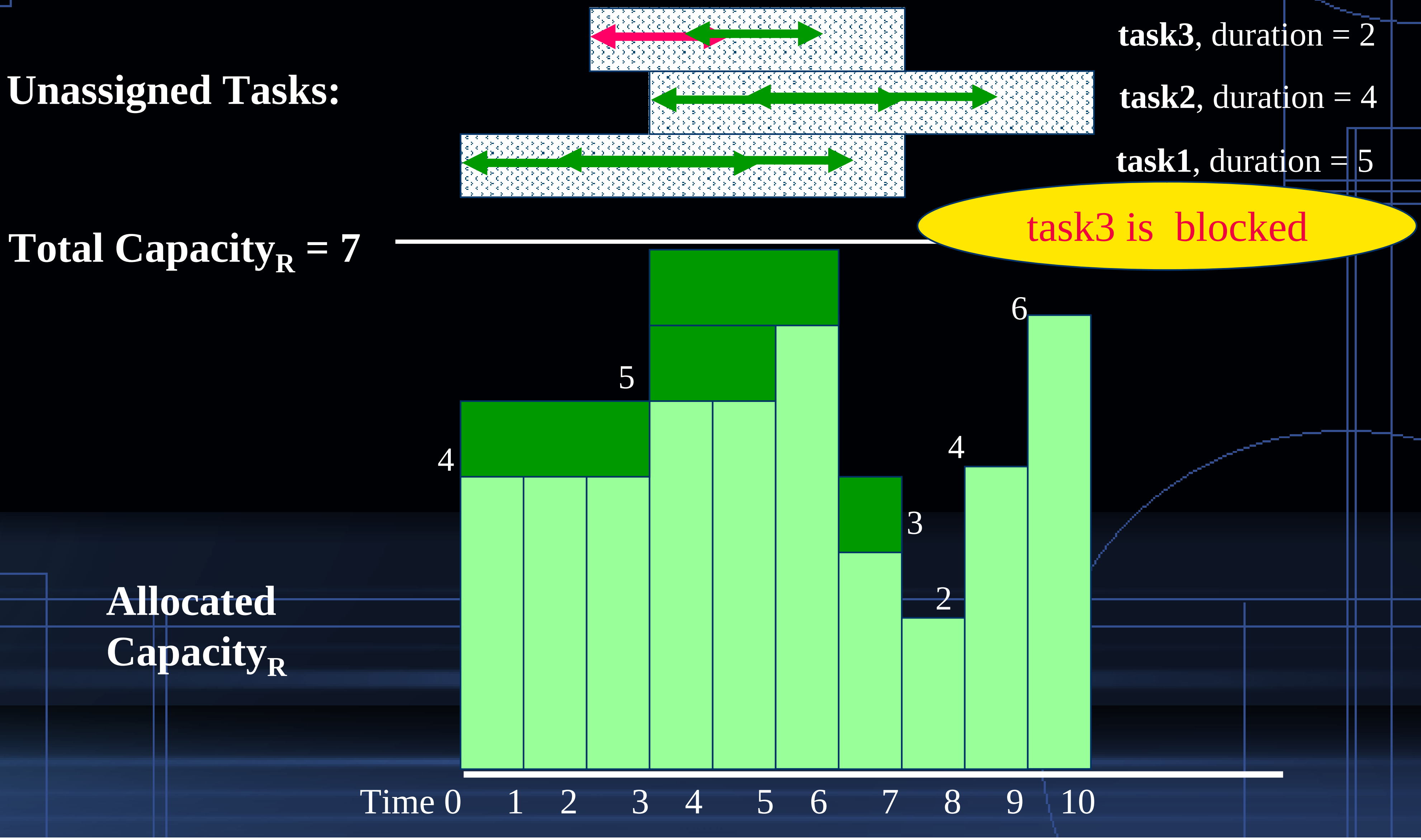
Alternatives for AMC domain:

- Relax Constraints
  - Delay
  - Overallocate
  - Bump
- Rearrange existing schedule to insert unassignable missions (e.g. ‘*Task Swap*’)
  - Must preserve feasibility and priority
  - Must be computationally tractable
- Generate new mission ordering and reschedule (e.g. ‘Squeaky Wheel Optimization’)

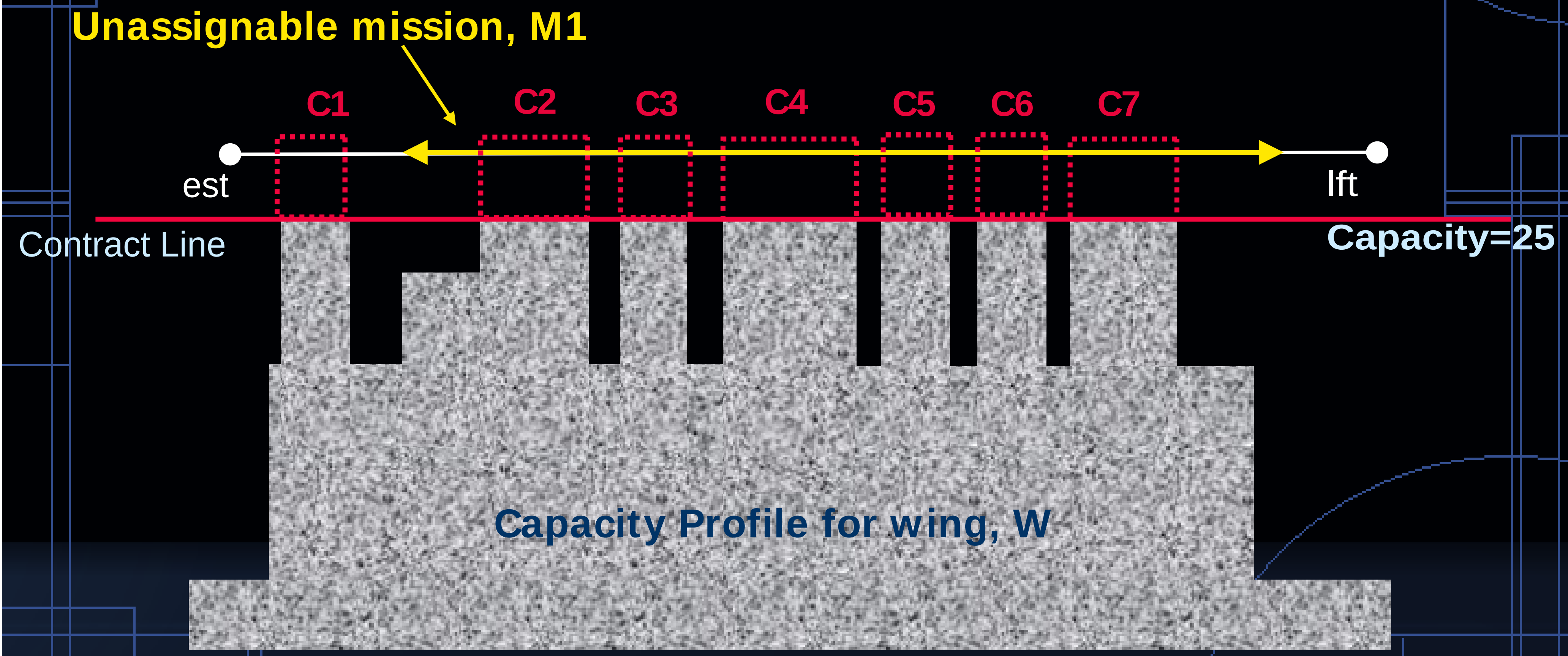
Schedule Construction: Some Fundamentals

Task Insertion -A greedy approach

(Earliest Start Time Assignment)



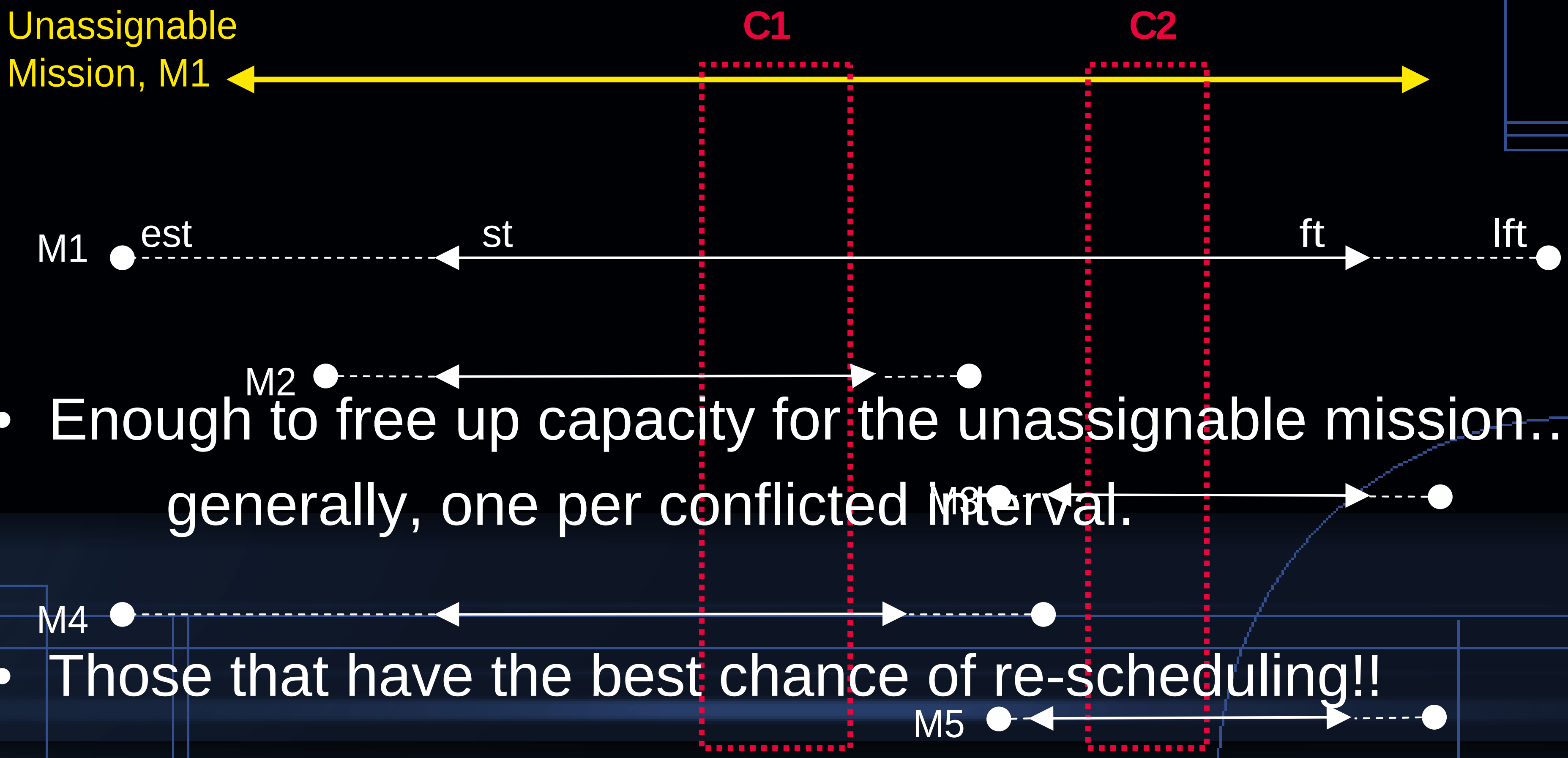
Scheduling an unassignable task M1 depends on reducing capacity in conflicted regions.



Over-subscribed scheduling (with preferences)

1st: Un-schedule conflicted missions *(TaskSwap)*

-but which missions should be unscheduled?

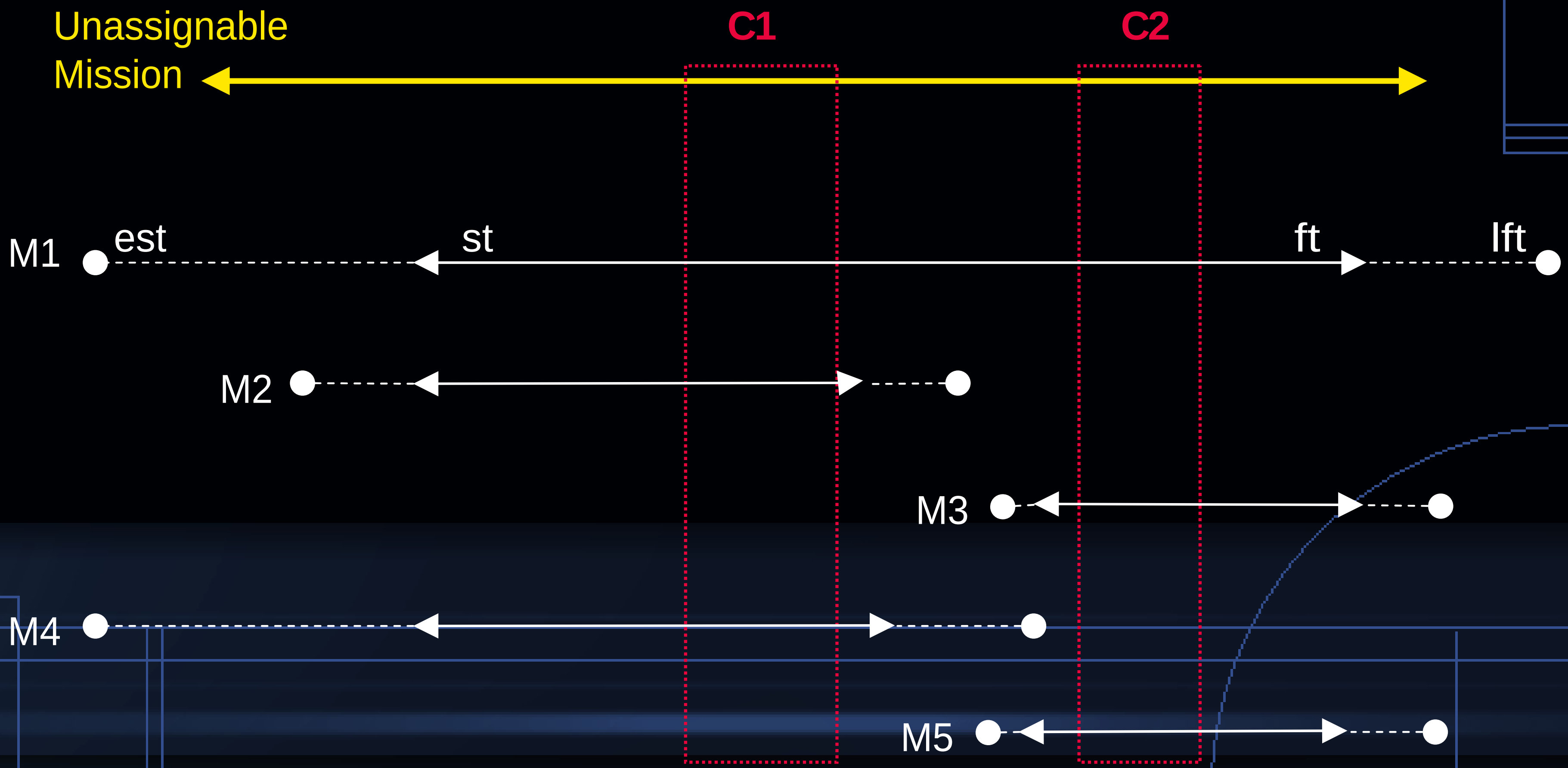


- Enough to free up capacity for the unassignable mission… generally, one per conflicted interval.
- Those that have the best chance of re-scheduling!!

Over-subscribed scheduling (with preferences)

So, which missions have the best chance of re-scheduling?

-Those that have the most flexibility.



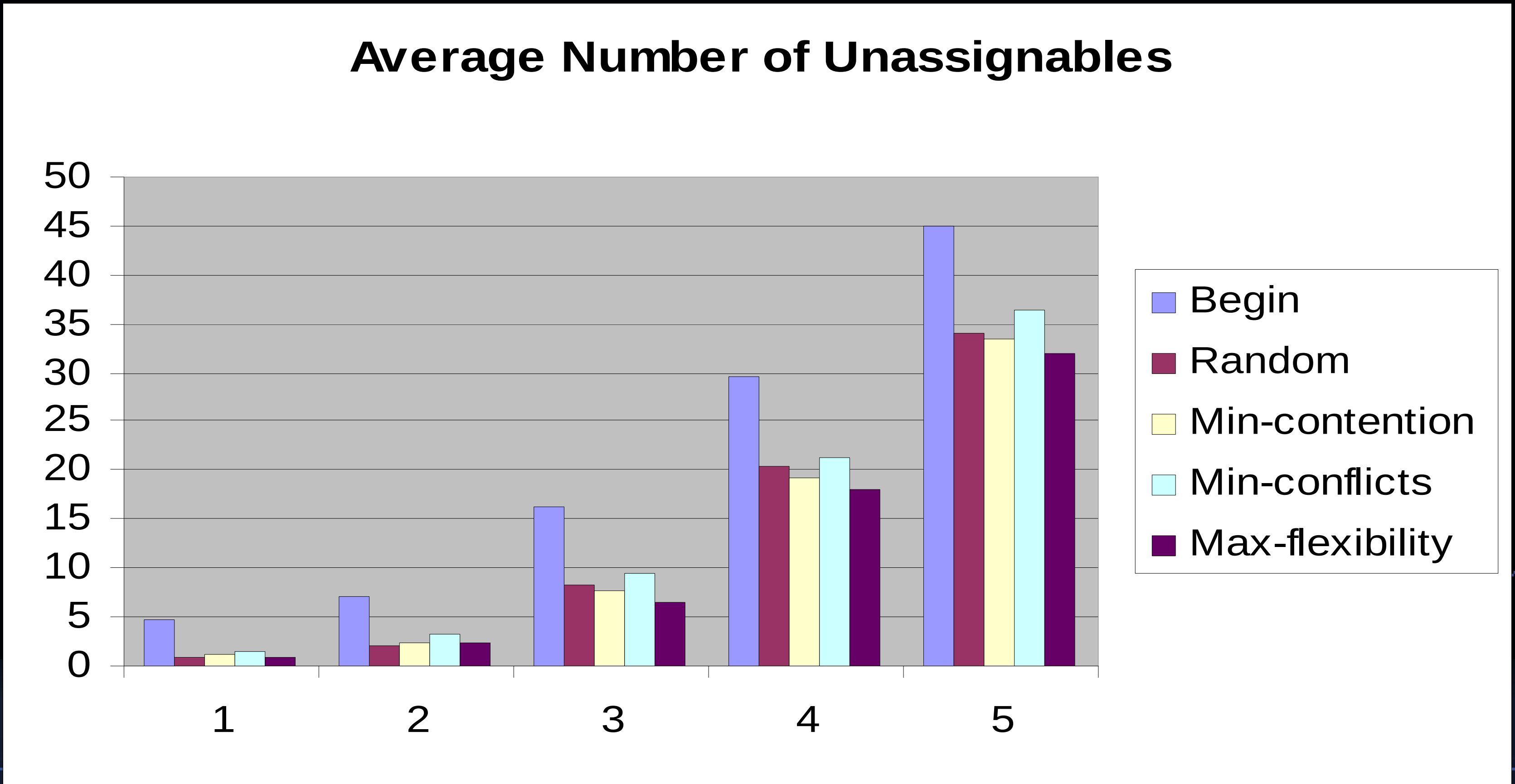
Over-subscribed scheduling (with preferences)

Heuristics for Maximum Flexibility

- Max-flexibility = mission-run-time/feasible-window-size
  - Inexpensive to compute
  - Doesn’t take into account contention with other missions
- Min-conflicts = count intervals that are at-capacity which conflict with the feasible window.
  - More expensive to compute
  - More informed?
- Min-contention = conflict-count * conflict duration
  - Even more expensive to compute
  - More informed?

Over-subscribed scheduling (with preferences)

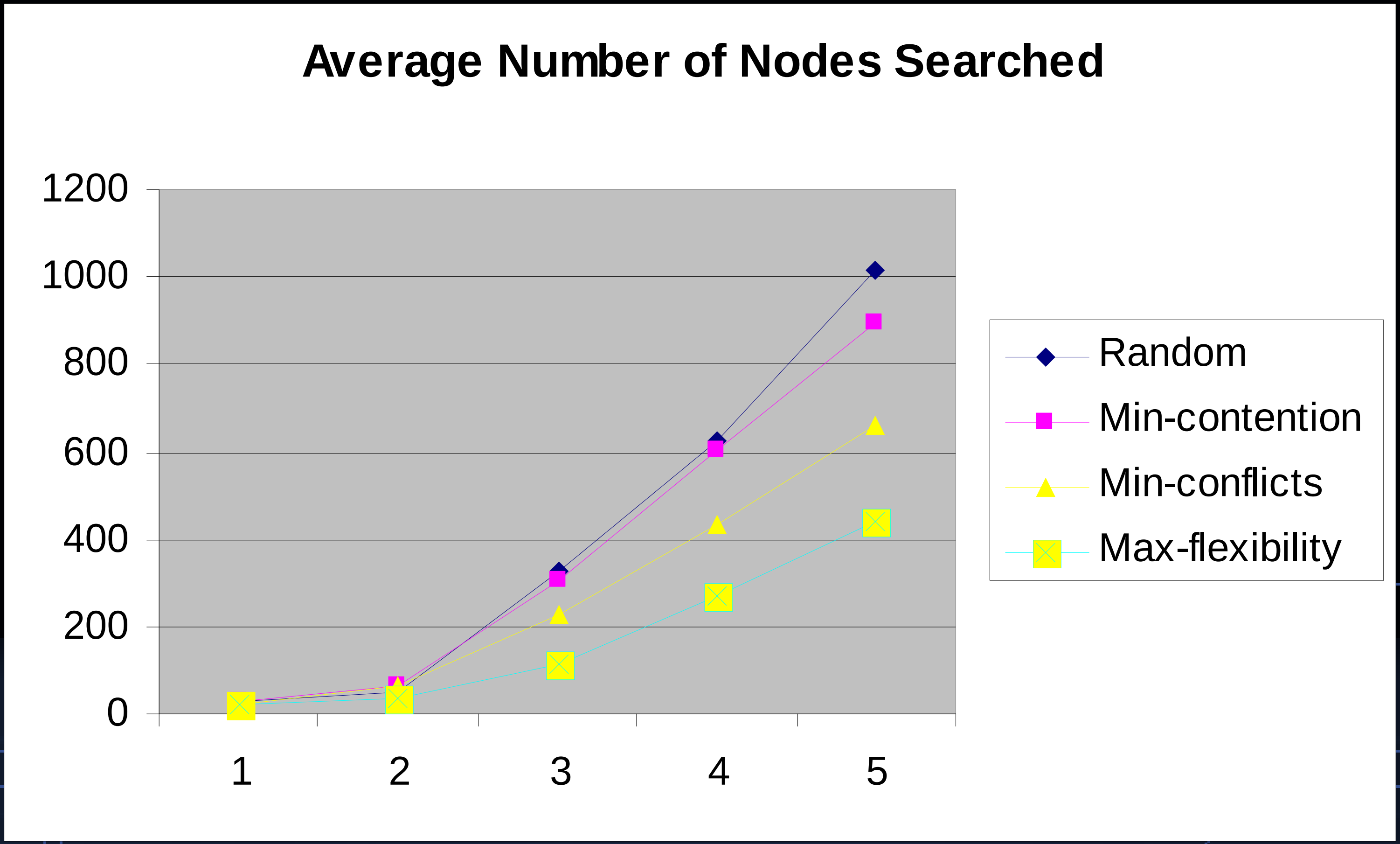
Max-flexibility dominates in solution quality



Max-flexibility assigned 42% of the unassignables. Min-contention, Random, and Min-conflicts assigned 38%, 36%, and 30%, respectively.

Over-subscribed scheduling (with preferences)

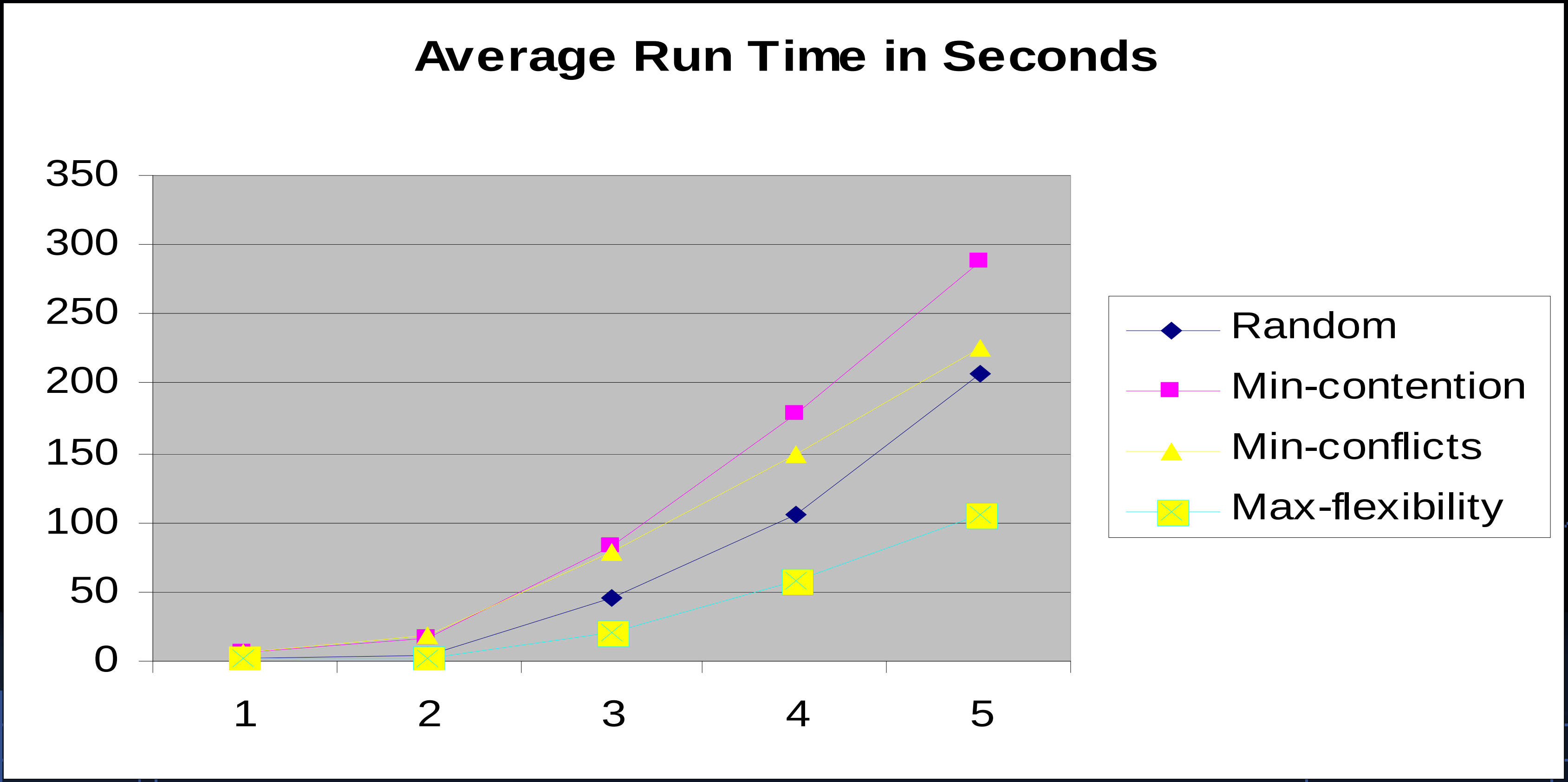
Max-flexibility generally searches the fewest nodes



Random selection is the worst.

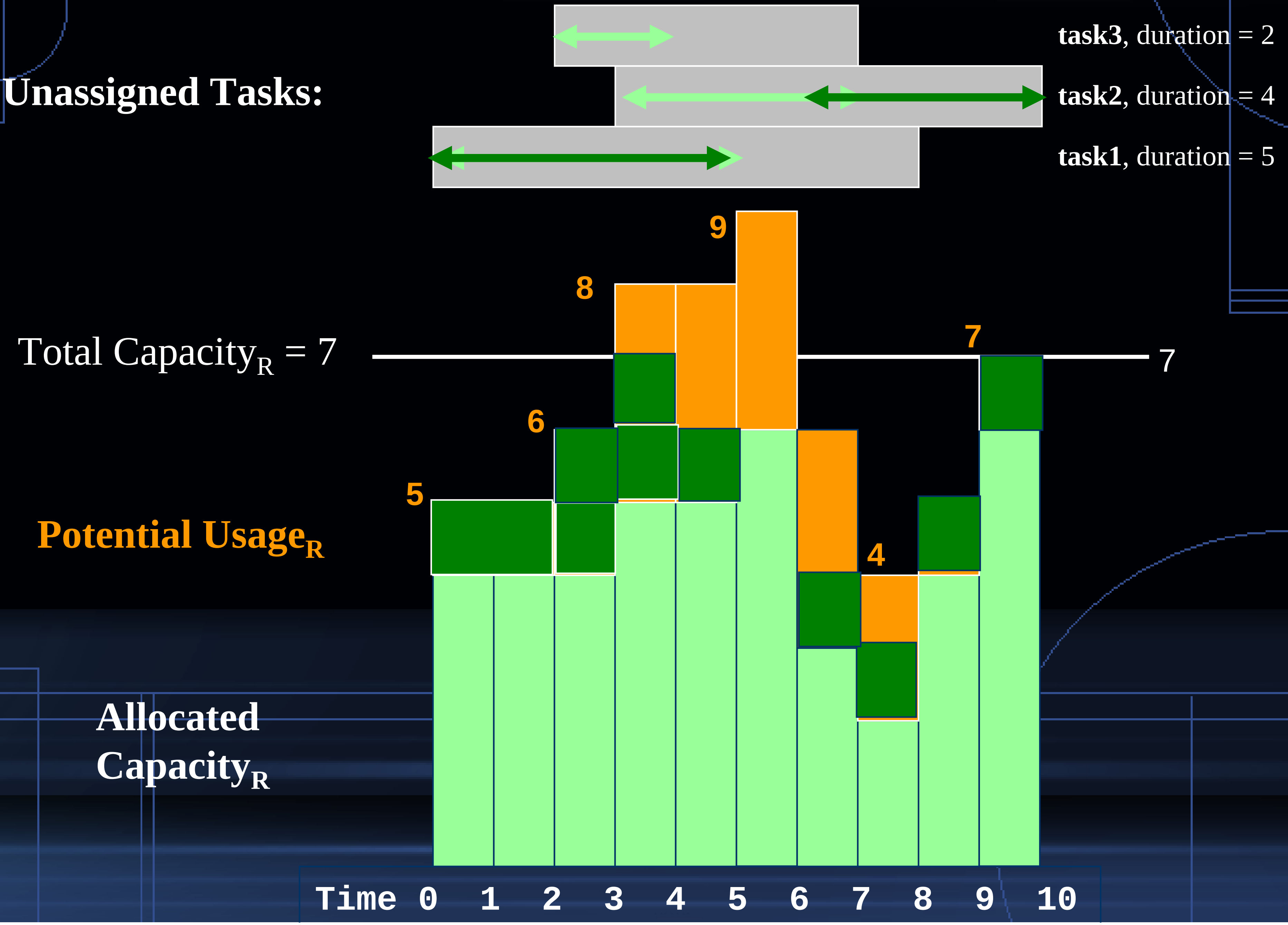
Over-subscribed scheduling (with preferences)

Max-flexibility dominates in average run-time.

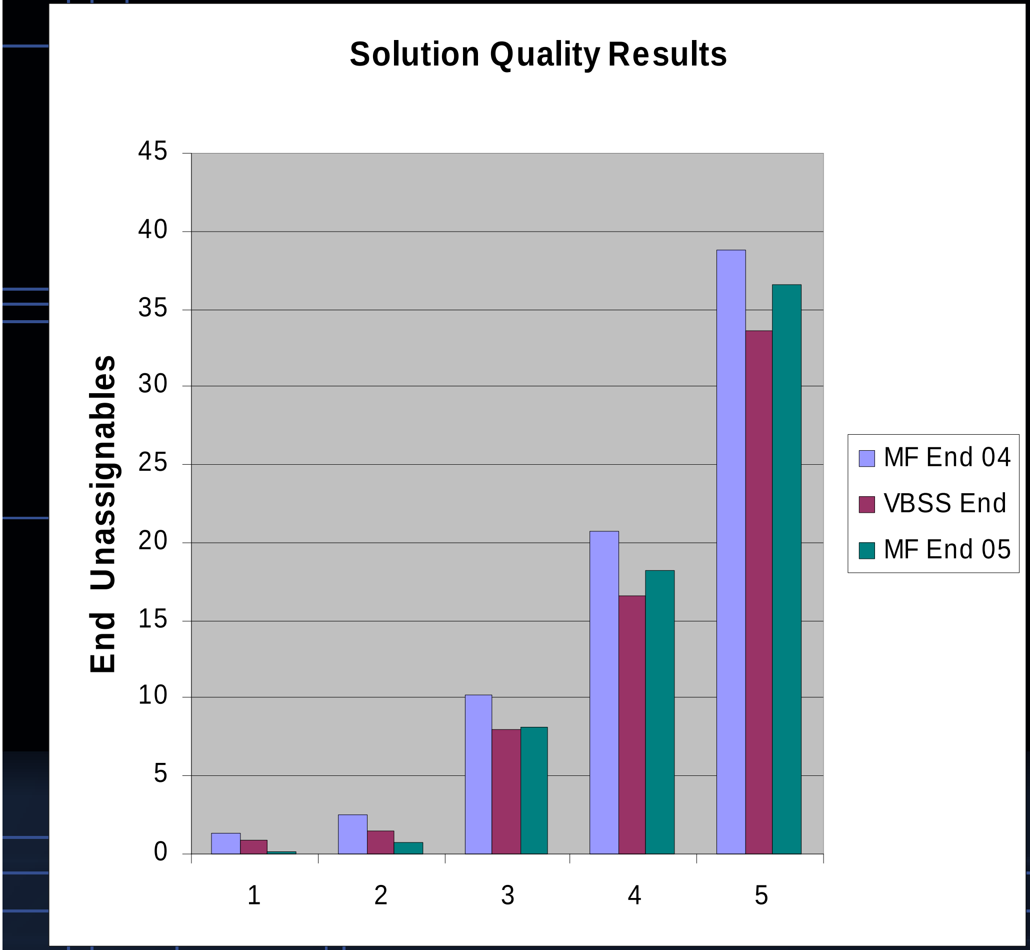


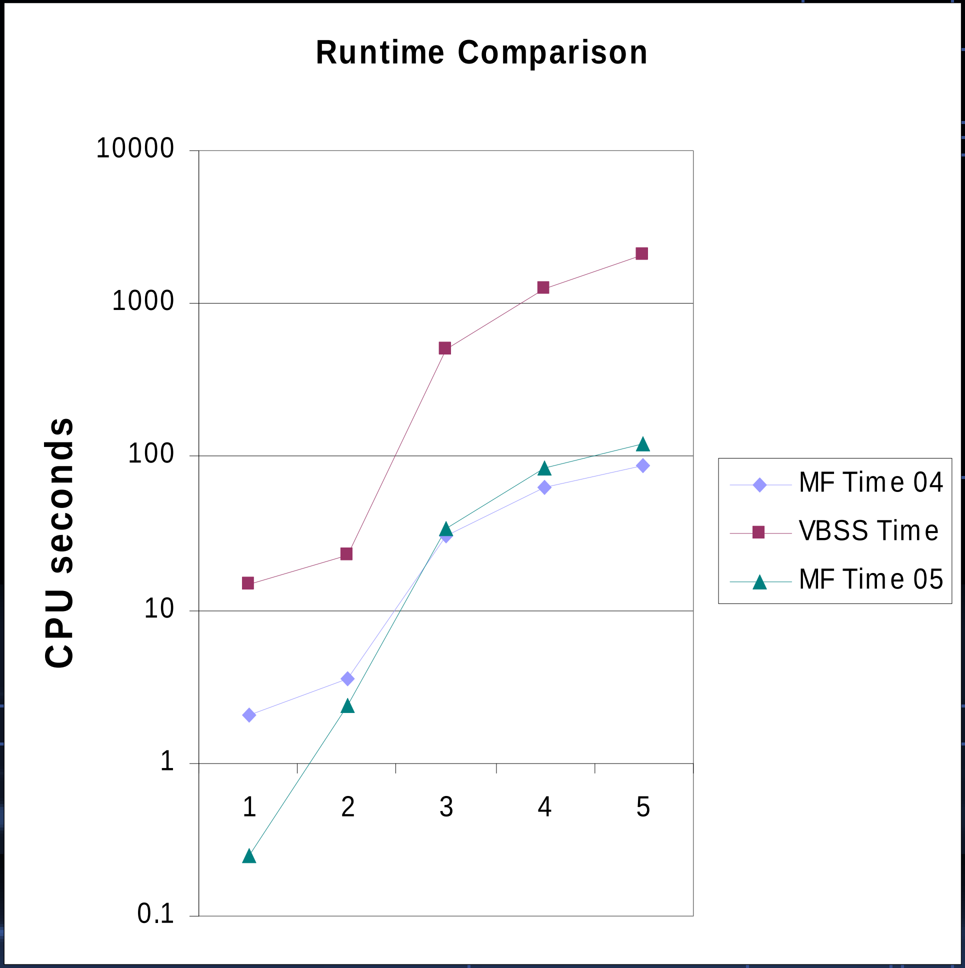
Min-contention is the worst.

Schedule Construction:
More Informed Task Insertion: Max-Availability



TaskSwap with pruning vs. TaskSwap, pruning, & MaxAv.





SWO: An Example

(Iteration Limit = 10; Move Operator = Fixed distance, 2)

- Initial task ordering (permutation): (T7 T5 T1 T2 T4 T3 T6)
- Build schedule; → **T1, T4, T3** *not* assigned (Squeaky Wheels).
- New task permutation, after moving squeaky wheels forward 2:

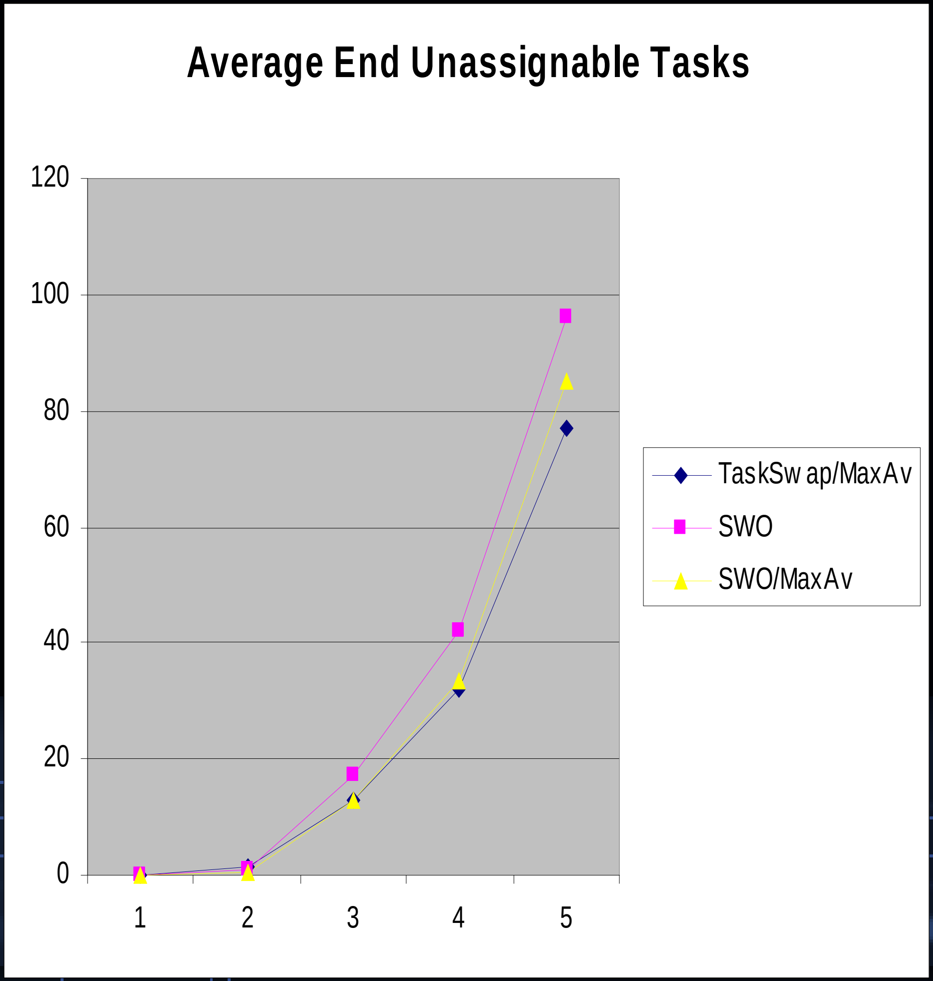
  (**T1** T7 **T4** **T3** T5 T2 T6)
- Build schedule; → **T4, T5** not assigned.
- New task permutation, after moving squeaky wheels forward 2:

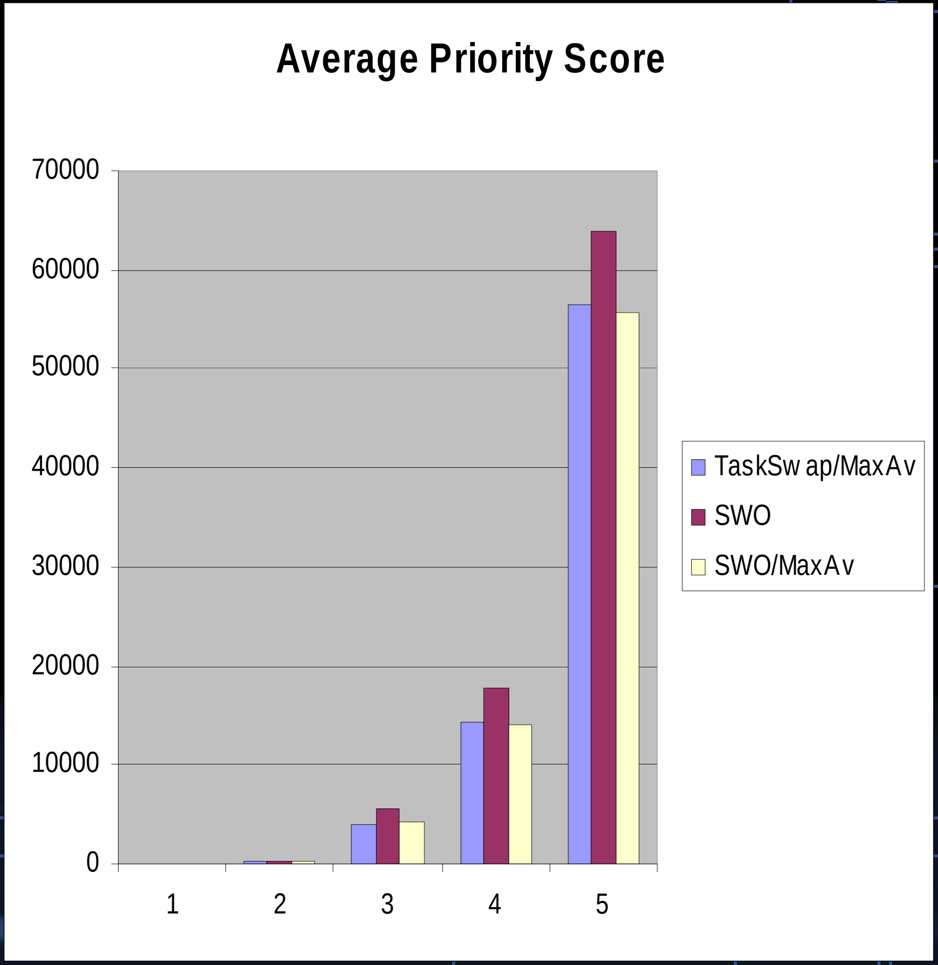
  (**T4** T1 **T5** T7 T3 T2 T6)
- Build schedule; → **T1** not assigned.
- New task permutation, after moving squeaky wheels forward 2:

  (**T1** T4 T5 T7 T3 T2 T6)
- Build schedule; All tasks assigned successfully; END.

Over-subscribed scheduling with preferences:

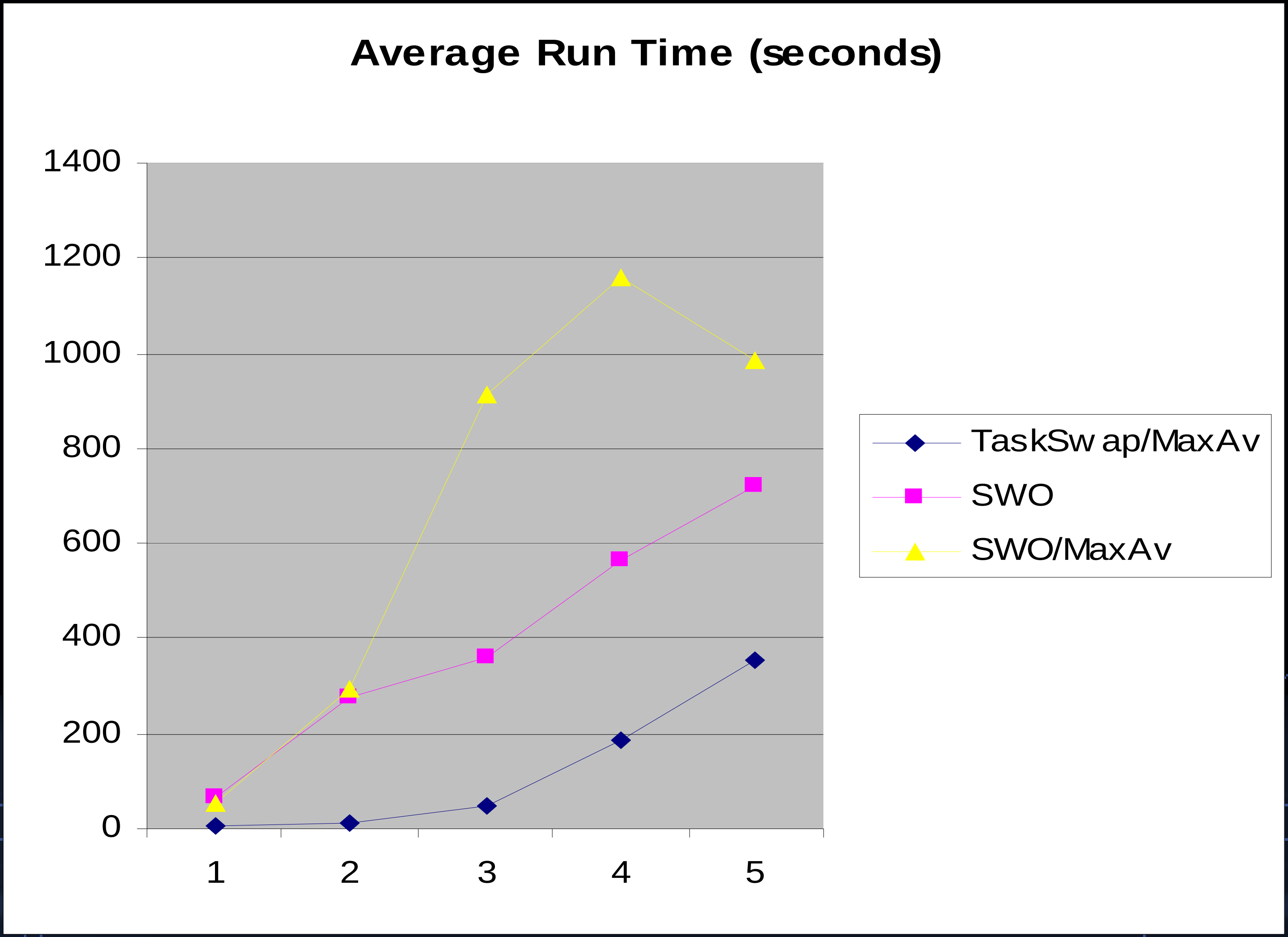
Scheduling Space (TaskSwap) vs. Permutation Space (SWO) *-AMC domain*





Over-subscribed scheduling with preferences:

TaskSwap & SWO: Run Time



Air Force Satellite Control Network (AFSCN) Access Scheduling

Input:

Request1:

Download data from satellite1 to ground-station1 in time window W.

Request2

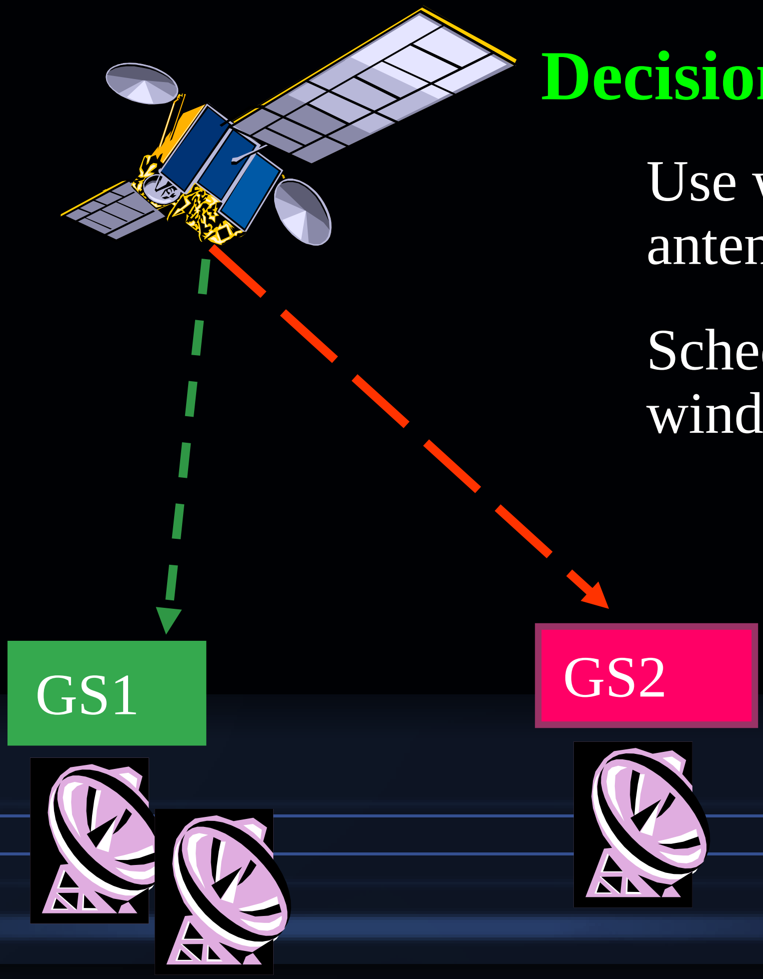
…

Request-n

Decisions:

Use which ground station and antenna?

Schedule in which time window?



Over-subscribed scheduling (with preferences)

Domain characteristics: Air Force Satellite Control Network

1. Tasks are constrained to varying degrees.
2. Most tasks can be assigned on more than one resource.
3. Task duration changes depending on resource assignment.
4. No explicit task priority specified.
5. *more here…*

Would the TaskSwap techniques apply?

Domain Comparisons: AFSCN and AMC

The main difference is that of task priority; other differences are of degree:

	AFSCN	AMC
Hard Priority Constraint?	No	Yes
Number of Tasks	419-483	983
Resource Capacity	1-3	4-37
Average Temporal Flexibility (task duration/window size)	(0.69968086 - 0.7595485)	0.49961752

Squeaky Wheel Optimization vs. TaskSwap

- Tested on five days of actual AFSCN data.*
- SWO iteration limit = 500;
- For unassignable tasks, *u*, *MoveDistance*(*u*) = 5.
- Both TS and SWO use max-availability to construct schedules.
- Objective: minimize end number of unassignable tasks.

Results: SWO outperforms TS.

Problem	Initial Unassignables	End SWO	End TS
R1	**58**	45	**49**
R2	**38**	30	**34**
R3	**27**	18	**20**
R4	**37**	28	**32**
R5	**19**	13	**15**

Two techniques: Strengths and Weaknesses

- Both TS and SWO start from a good initial schedule.
- SWO permutation-space search is blind to schedule state.
  - Scheduling/unscheduling of many tasks, allows jumping to promising problem spaces.
  - Good initial solution + good move operator = quick convergence.
  - Undirected search may lead to thrashing and cycling after good solution reached.
- TS, designed for schedule stability, examines schedule state closely with task retraction and insertion heuristics.
  - Moves in a directed way from a good solution to a better one.
  - May miss the “best” solution due to the inability to undo a good solution.

Reported in AAAI-07 :

Combining Strengths in a Hybrid*

(Kramer, Barbulescu, Smith)

- Problem: both techniques get stuck in a good solution.
- Solution: combine SWO and TS in a coarse-grained hybrid.
- Define a parameter, ?, iterations without improvement.
- Run SWO, and if ? is reached, run TS (*HybridSWO*).
- Similarly for TS, then SWO (*HybridTS*).

Over-subscribed scheduling with preferences:

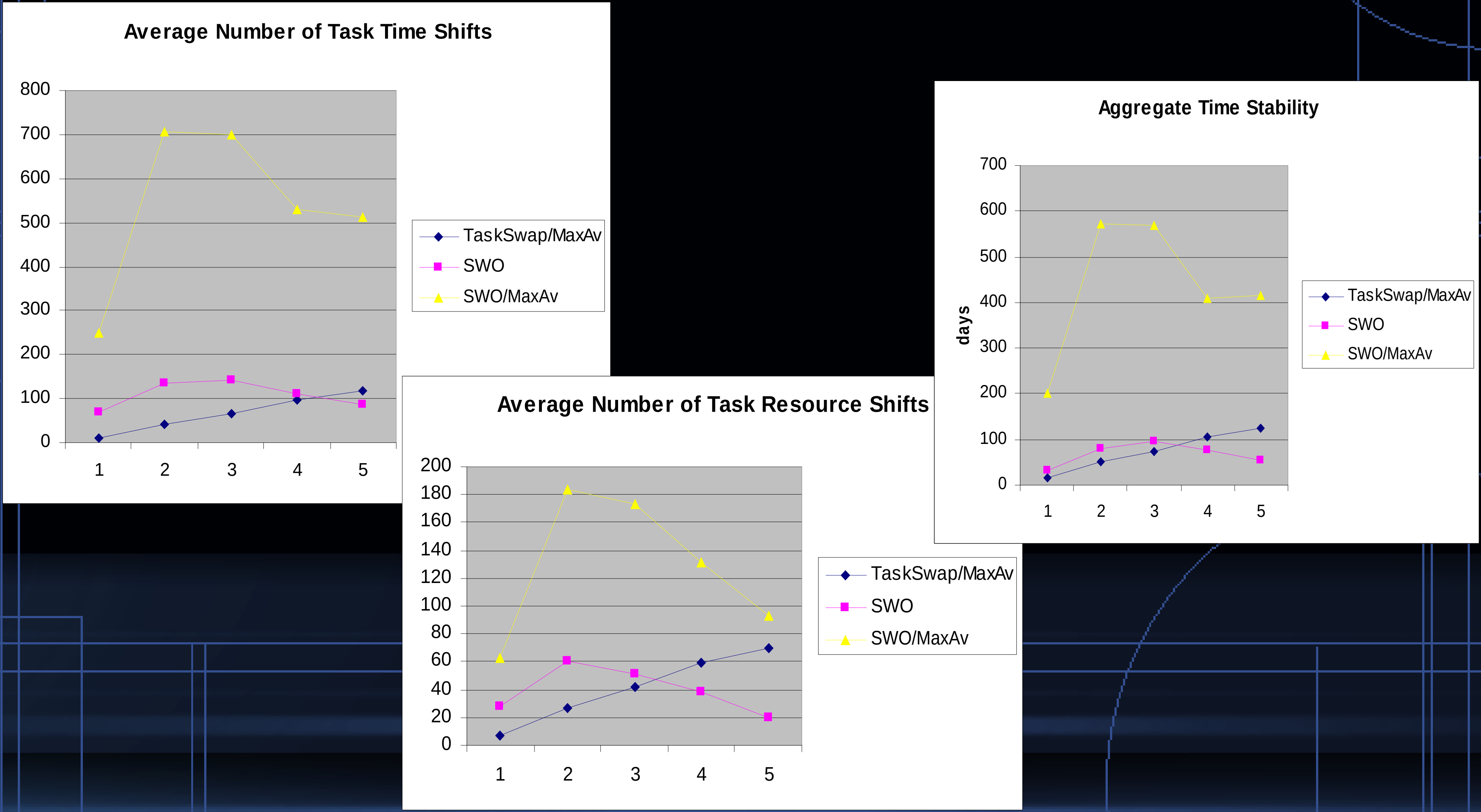
Preference for Schedule Stability

Why care about stable schedules?

- Because human’s care (mixed initiative)
- Reduce cognitive load in understanding fluctuating schedules
- Context switching cost
  - Hedge against an incomplete modeling
- Reduced communication (distributed scheduling)

Over-subscribed scheduling with preferences

Stability: *Task Swap* vs. *SWO*



Average Number of Task Time Shifts

Average Number of Task Resource Shifts

Aggregate Time Stability

Caveat: SWO sometimes swaps tasks out of the schedule.

Over-subscribed scheduling with preferences:

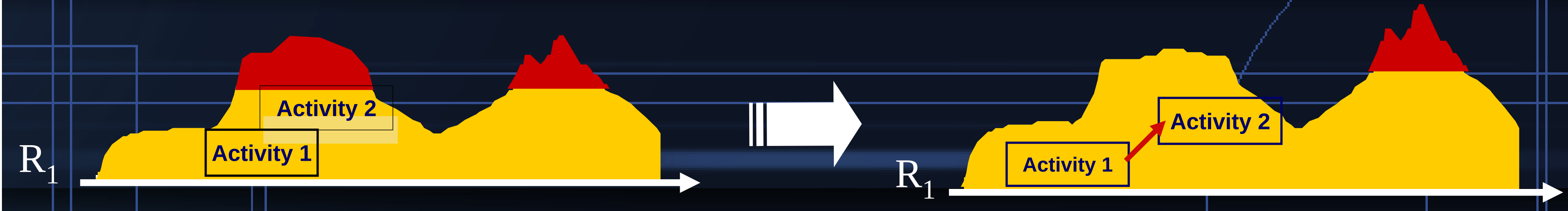
Preference for Schedule Robustness

Hedging Against Possibility of Unexpected Task Behavior

Perspective: *Build schedules that retain flexibility and can absorb some amount of unpredictability in execution*

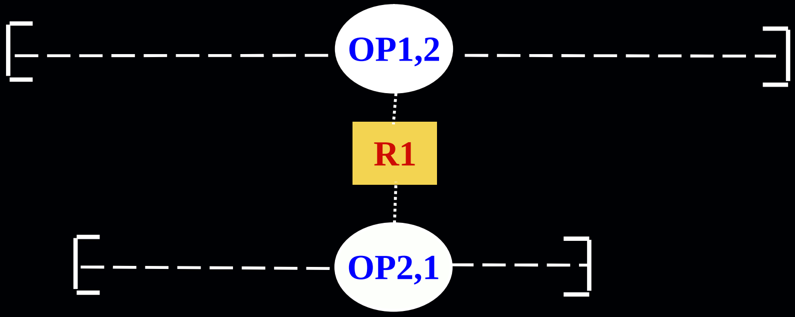
- task execution windows instead of precise times
- resource options instead of precise assignments
- process redundancy to increase likelihood of success

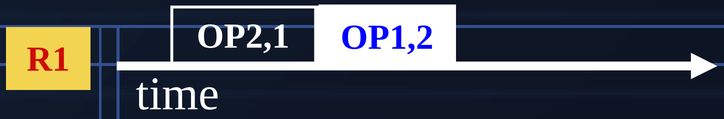
Basic Approach: *Partial-order scheduling procedures*

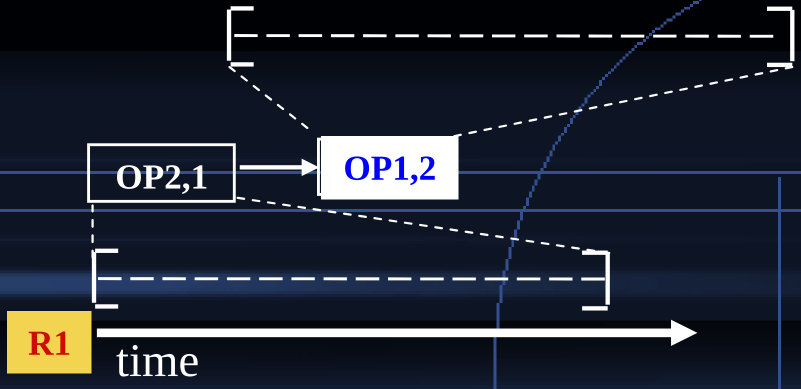


“Flexible Times” Schedules

Sequence activities that are competing for the same resources - *let start and end times float*



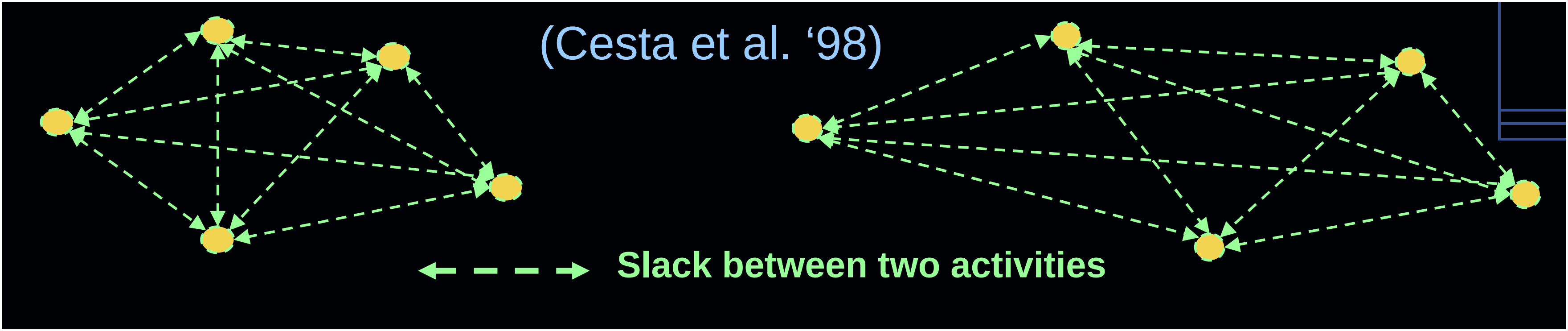




Characterizing Schedule Robustness

- **Fluidity:** *average slack in the schedule*

$$rb = \sum_{h \neq l} \frac{|d(e_{ah}, s_{al}) - d(s_{al}, e_{ah})|}{H \times N \times (N-1)} \times 100$$

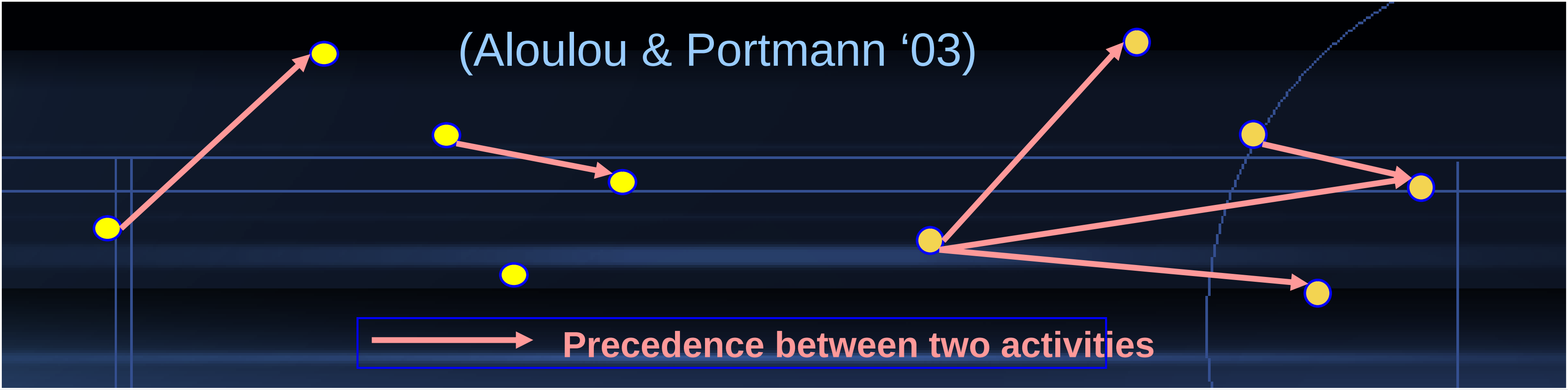


(Cesta et al. ‘98)

Slack between two activities

- **Flexibility:** *average nbr. of ordered activities*

$$flex = \sum_{i=1}^{N} \sum_{\substack{j=1: \\ dij \times dji < 0}}^{N} \frac{1}{N(N-1)}$$



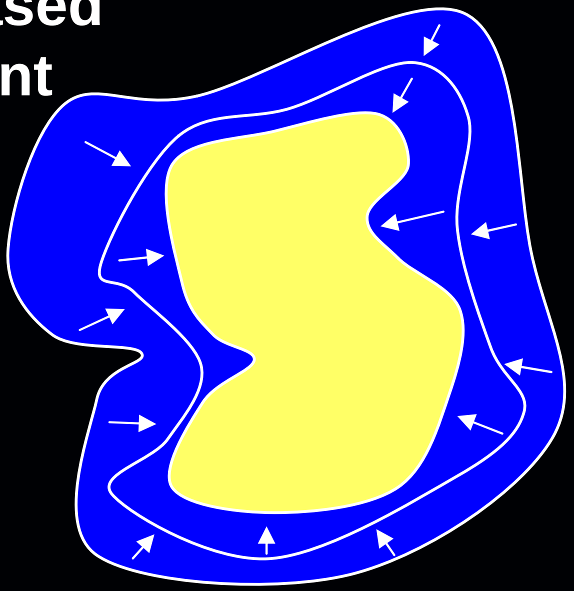
(Aloulou & Portmann ‘03)

Precedence between two activities

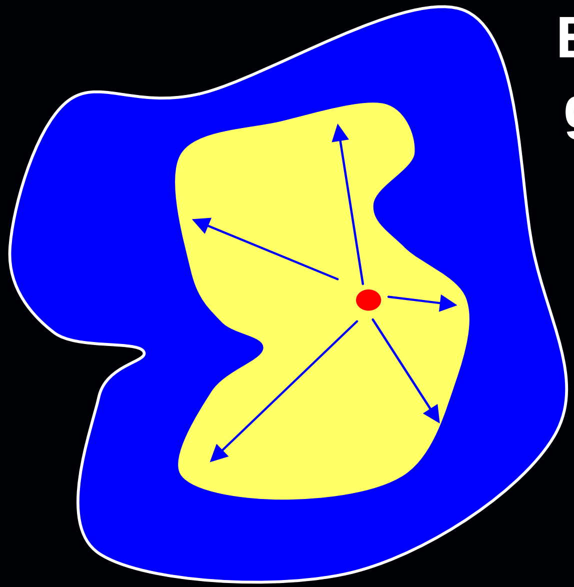
Building Robust Schedules: A Counter-Intuitive Result

[Policella, Smith, Cesta, Oddi - 2004]

Envelope-based commitment (EBA)



EST solution + generalization ($ESTA^C$)



$ESTA_C$ solution procedure generally dominates

Over-subscribed scheduling (with preferences)

Some Lessons Learned

The types and levels of soft constraints in the target problem often determines most effective approaches

- For problems without task priority, the more disruptive SWO permutation-based search is more effective.
- For problems with task priority, as oversubscription increases, a repair-based search such as TaskSwap is better in traversing the constrained space.

Soft constraints and preferences: Reasoning Over Quality & Utility

Classical scheduling problem performance objectives:

- meeting due dates
- maximizing throughput

Quality-oriented constraint considerations:

- variants on prioritized tasks
  - (Zweben, et. al., 1994 Scheduling and re-scheduling with iterative repair), …*more* TBA.….
- simple task quality models *(…refs TBA)*

Evolution towards more complex quality models:

- Increasing schedule quality with service time on-task
- design-to-time scheduling (Garvey & Lesser, 1993)
- imprecise computation (Liu et al. 1994)
- IRIS (Increasing Reward with Increasing Service) Dey, Kurose, & Towsley, 1996),
- progressive processing (Mouaddib & Zilberstein, 1998)
- anytime scheduling (Schwarzfischer, 2004)
- incremental scheduling to maximize quality (Gallagher, Smith, Zimmerman, 2006)

HTN models of quality accumulation

- taems *(…refs TBA)*
- ctaems

Increasing schedule quality with service time on-task

‘KIDS’ Knowledge-Intensive Dynamic Systems

… where the production and manipulation of knowledge products is driven by the goal of maximizing process quality…

…and the quality (or utility) gained by executing a given activity is a function not *only* of ascribed worth (priority), but also of the time and resources allocated to it…

…and there is generally more to do than can be accommodated given the available resources and deadline constraints…

…and the processes are unpredictable in their outcomes and require continual dynamic adjustment and revision.

Practical KIDS problems:

Typically large-scale, multi-actor systems where time-on-task and skill level play key roles in planning and execution of high quality production processes:

– Quality testing of large software systems
  - Confidence that a component is robust varies with time spent testing
  - Delivery deadlines force choices as to level of testing

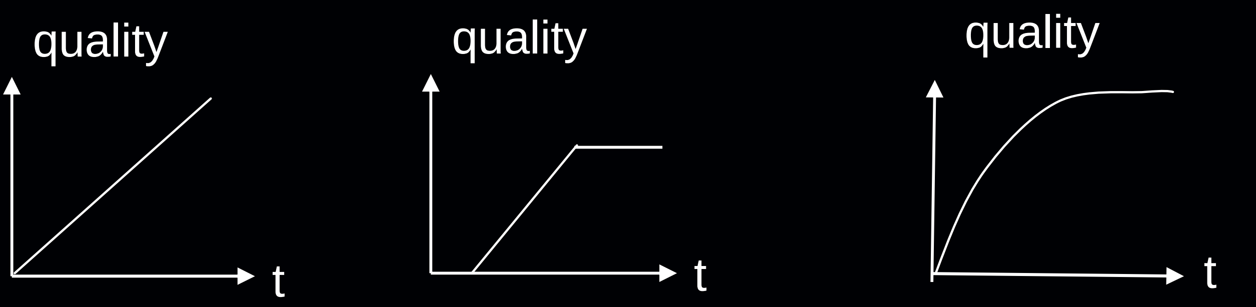
– Planetary exploration
  - Time spent by robotic rovers on sampling and transit heavily impacts scientific return
  - Ongoing analysis continually reprioritizes and augments tasks

– CNN, AP news services
  - News report contribute to overall program quality based on event importance, time spent on-site & in-production, and reporter skill level
  - At any time there are more news events than resources can cover -- new events arise unpredictably and continually.

Scheduling to Maximize Quality

Quality profile:

quality is a nondecreasing function of time



(a) Linear

(b) Piecewise Linear

(c) Nonlinear

Examples: telescopes, cameras/videos, anytime optimization algorithms running on a computer or a robot, knowledge intensive work

Domains: management of satellite operations, news reporting.

A quality-centric scheduling problem

Each activity *a1* in the agenda has:

priority *pr(a1)*, earliest start time *est(a1)* & latest finish time *lft(a1)*

A set of resources for assignment… each resource r_1 has:

- skill level $sk(r_1)$
- sequence-dependent setup time $setup(r_1,a_1,a_2)$ (to transition from activity a_1's state to activity a_2's)

Setup time does *not* contribute to problem quality…

Each activity requires exclusive use of a single resource and the quality garnered by executing *a1* on resource *r1* from *t1* to *t2* is:

1) $q(a1, r1) = pr(a1) \times sk(r1) \times dur(a1)$

dur(*a1*) = *t1* - *t2* … it's a free variable

A minimum req'd quality is enforced

Overall quality for *S* scheduled activities is:

2) $Q_S = \underset{k \in S}{S}\, q(a_k, r(a_k))$ where $r(a_k)$ is the resource assigned to a_k

Quality-Centric Scheduling

Representational Challenges

On the one hand…

Need to dynamically insert new activities into existing schedule over time argues for *flexible-times* scheduling

- Start & end times not anchored –float within time bounds
- Resource conflicts avoided by sequencing pairs of competing activities

..On the other hand..

Maintaining a schedule that *maximizes* quality is by definition a *fixed-times* schedule.

- Leaving duration flexible settles for lower quality than fully extending it…
- but higher quality solution may hide possibilities for better resource utilization and complicates accommodation of change

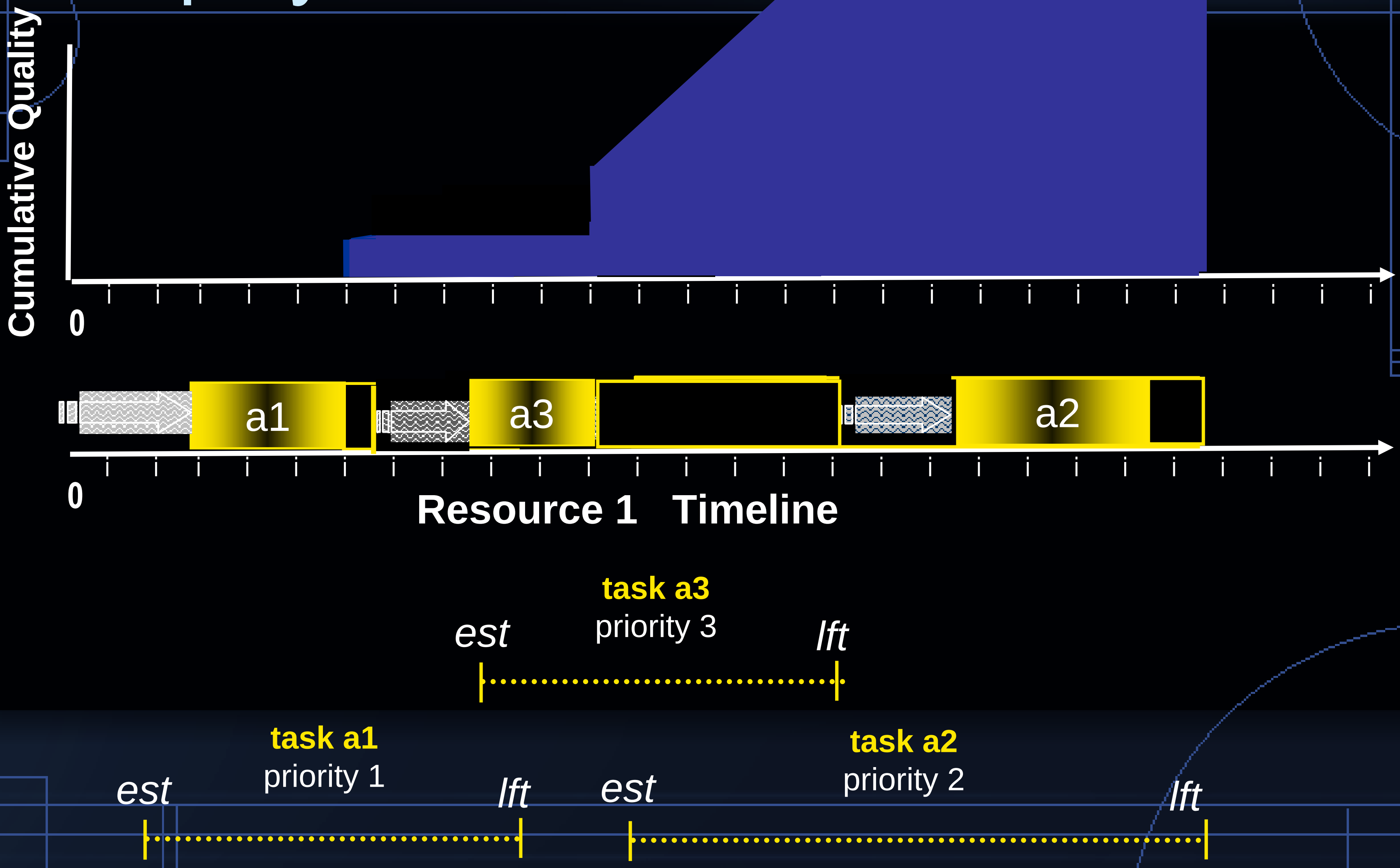
Approach: Dual Models of Schedule

- Flexible-times: Provides visibility of options
  - Employ a Simple Temporal Problem (STP) constraint network.
  - Only minimum duration on quality-accumulating activities are enforced (along with *est, lft, setup..)*
- Maintain a *fixed-time* representation of each resource's schedule (*quality timeline)*
  - Designates the start & end times for current sequence that yield 'sequence optimal' quality for the resource

Dual models allow us to *incrementally* maintain the sequence-optimal schedule as new activities come & go.

Building the Schedule:

Interplay of flexible & fi models



task a3
priority 3
est *lft*

task a1
priority 1
est *lft*

task a2
priority 2
est *lft*

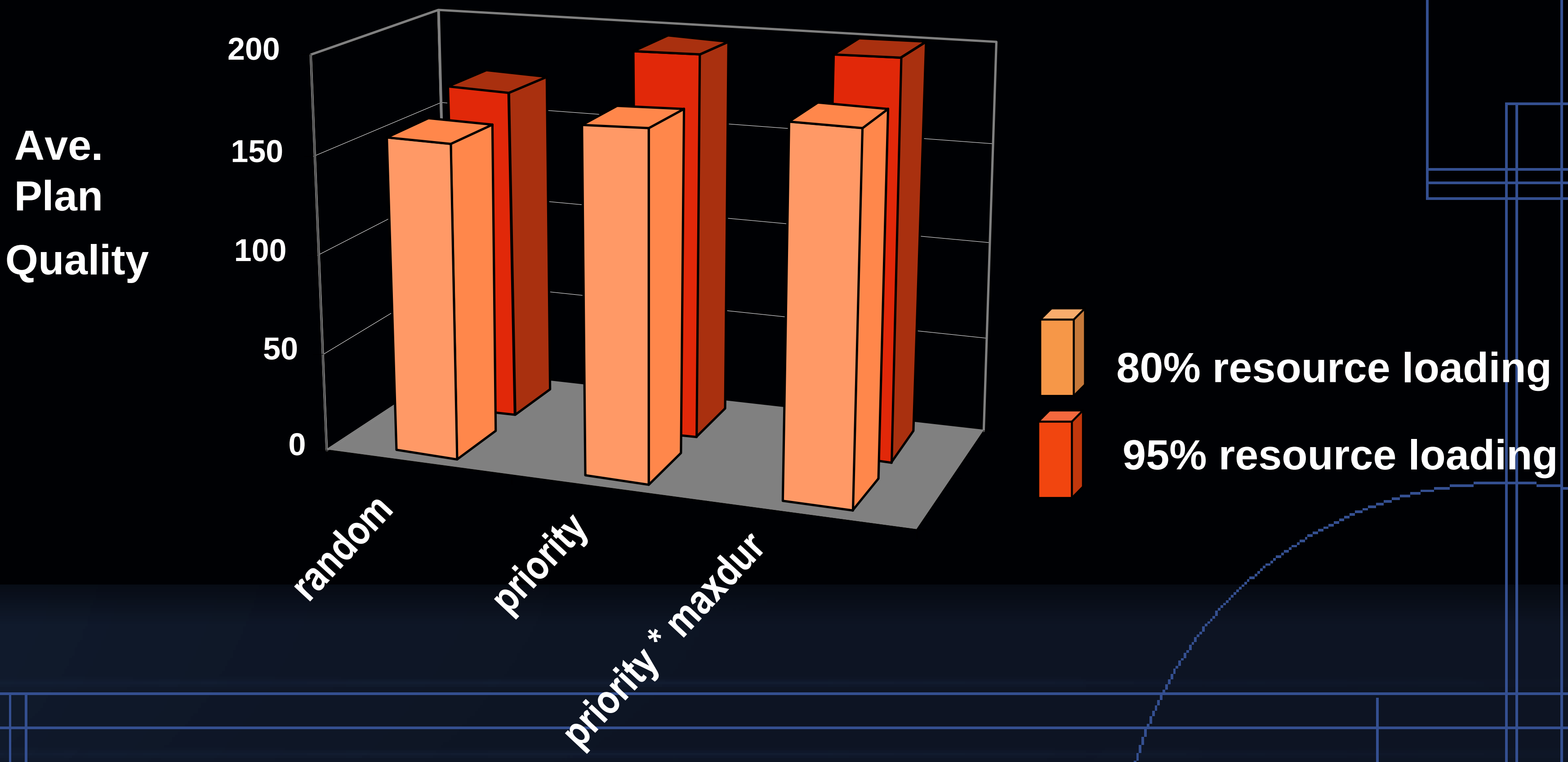
Algorithm:
Incremental Activity Insertion

Given an *agenda* of activities to insert in current *Schedule*:

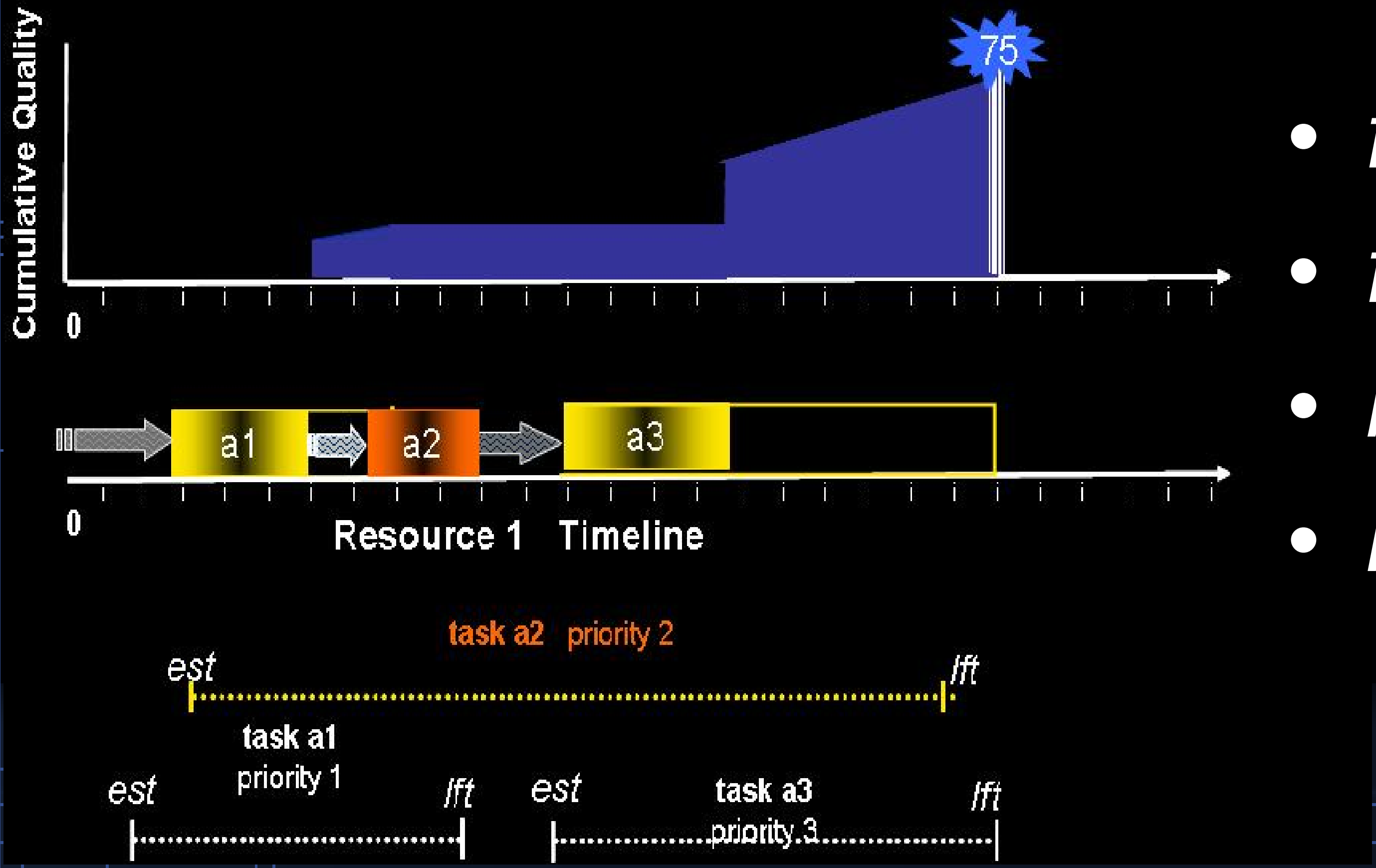
```
Activity ← Select-Activity(Agenda)

Schedule-Activity(Schedule; Activity)
1. Options ← Find-Options(Schedule; Activity)
2. Selected ← Choose-Option(Options)
3. Insert-In-Temporal-Network (Activity; Selected)
4. Insert-In-Quality-Timeline (Activity; Selected)
5. Return(Selected)
end
```

Increasing schedule quality with service time on-task

Activity Ordering Heuristics

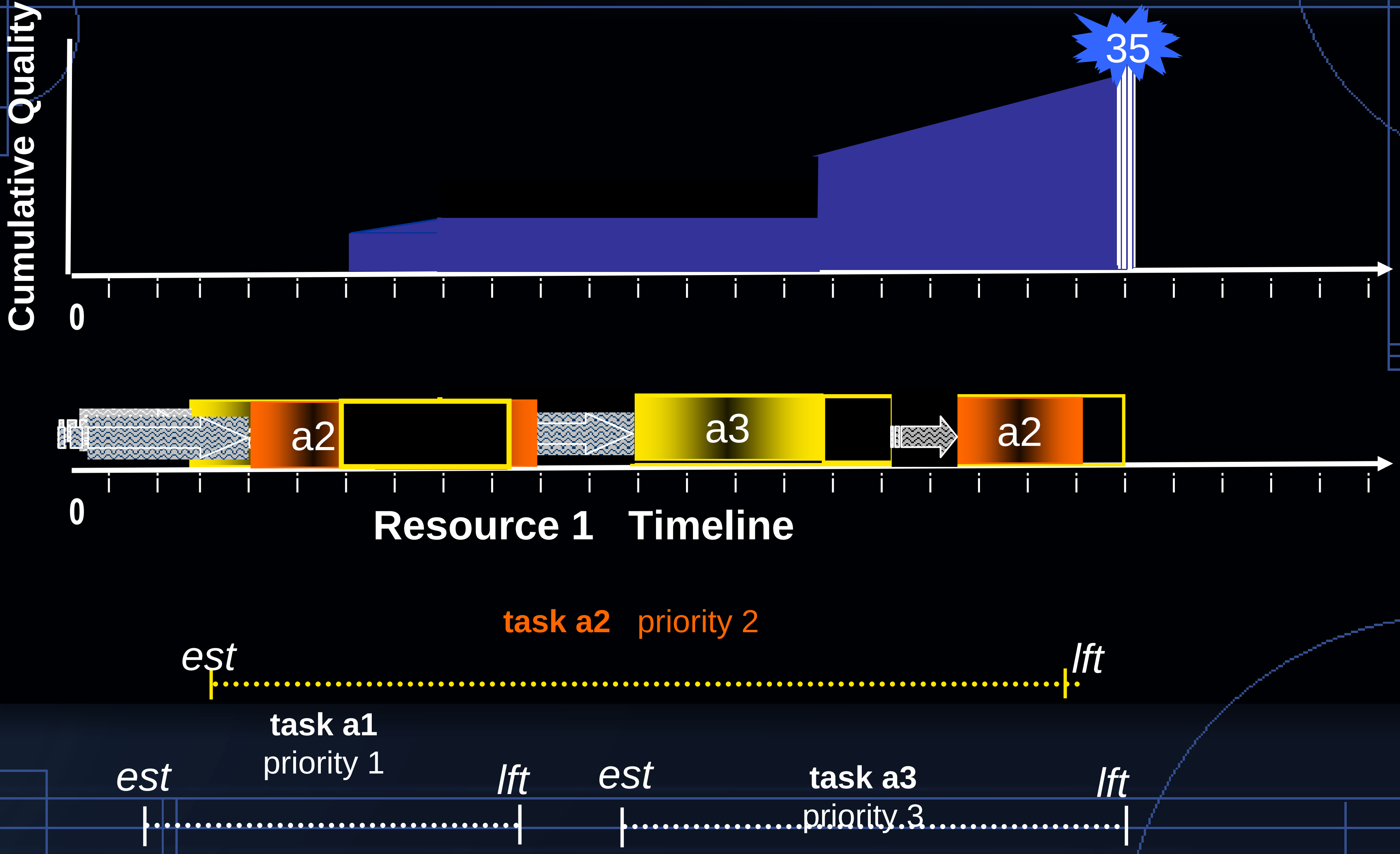


Activity Insertion Option Heuristics



- *first-slot*
- *first-gain-slot*
- *best-slot*
- *min-setup*

Generation of Insertion Options



Resource 1 Timeline

task a2 priority 2

est *lft*

task a1
priority 1

est *lft*

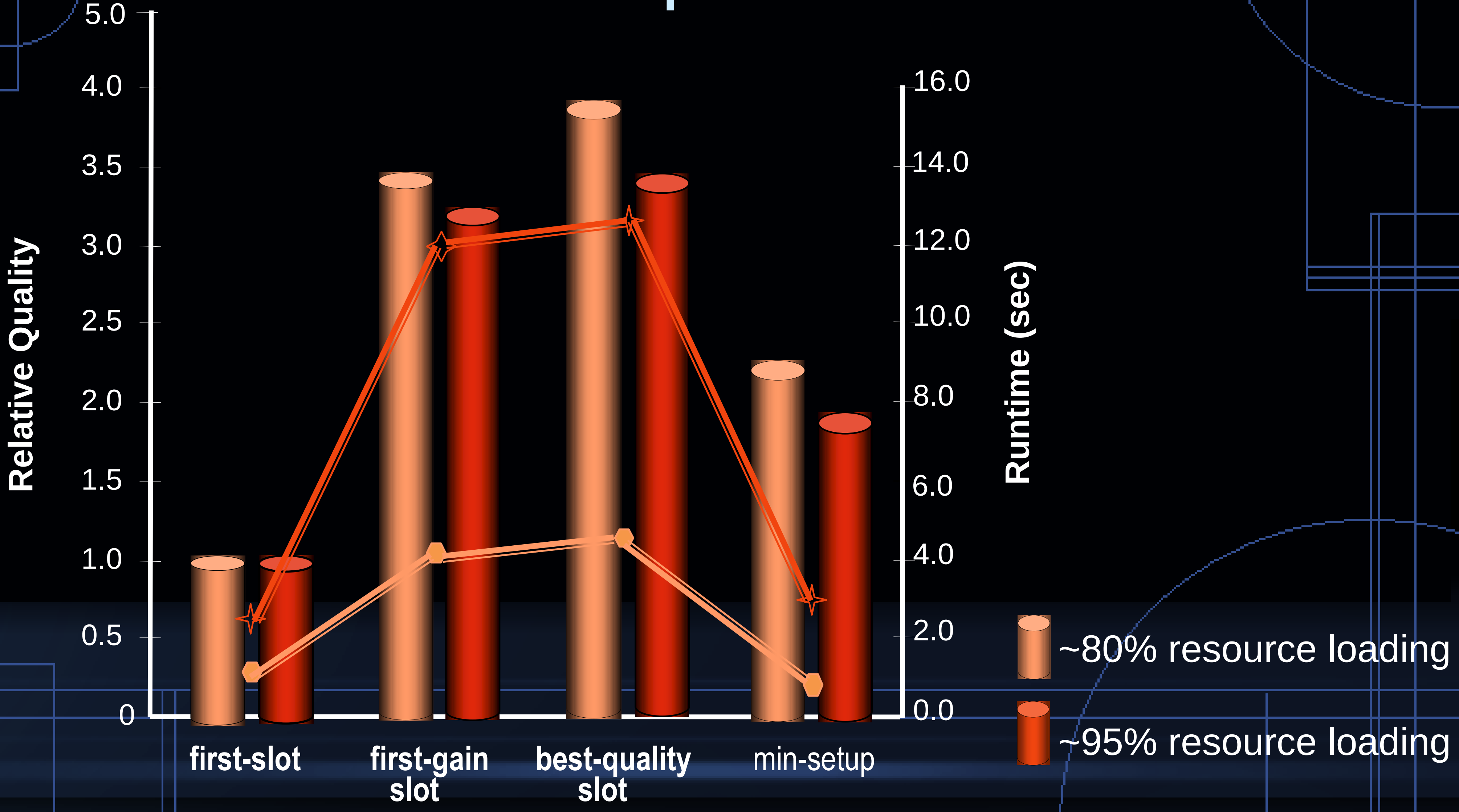
est

task a3
priority 3

lft

Increasing schedule quality with service time on-task

Insertion Option Heuristics



Experimental analysis:
A KIDS Domain.. “The News Agency”

- Agency covers prioritizes and covers newsworthy events so as to maximize the ‘quality’ of its programming product.
- News events have release times and deadlines & may arrive spuriously.
- Human resources (reporters) with skill levels – higher skilled reporter achieves a given report quality faster.
- Resources must be positioned at event location to conduct the report.
- Quality of a news report derives from event priority level, skill and duration of reporting effort.
- Each news report has a minimum quality requirement,

Experimental analysis: Problem test set generation

- 5 news event locations at various inter-city travel distances
- 3 reporters with random skill level (1 – 3) and random initial location
- News events with uniformly distributed priorities (1- 5) and minimum req'd quality of 1 (corresponds to 1 hr min duration for reporter of skill=1)
- 10 day time horizon
- Oversubscribed problems: 2 sets, 10 problems each
  - 100 events corresponding to ~20% free space on timelines
  - 200 events: ~5% free space (~50% of activities scheduled)
- 10 randomly generation new events then sequentially given to the scheduler…

Incremental Scheduling Strategies

i. Simple' insertion of event in available slot on resource timeline

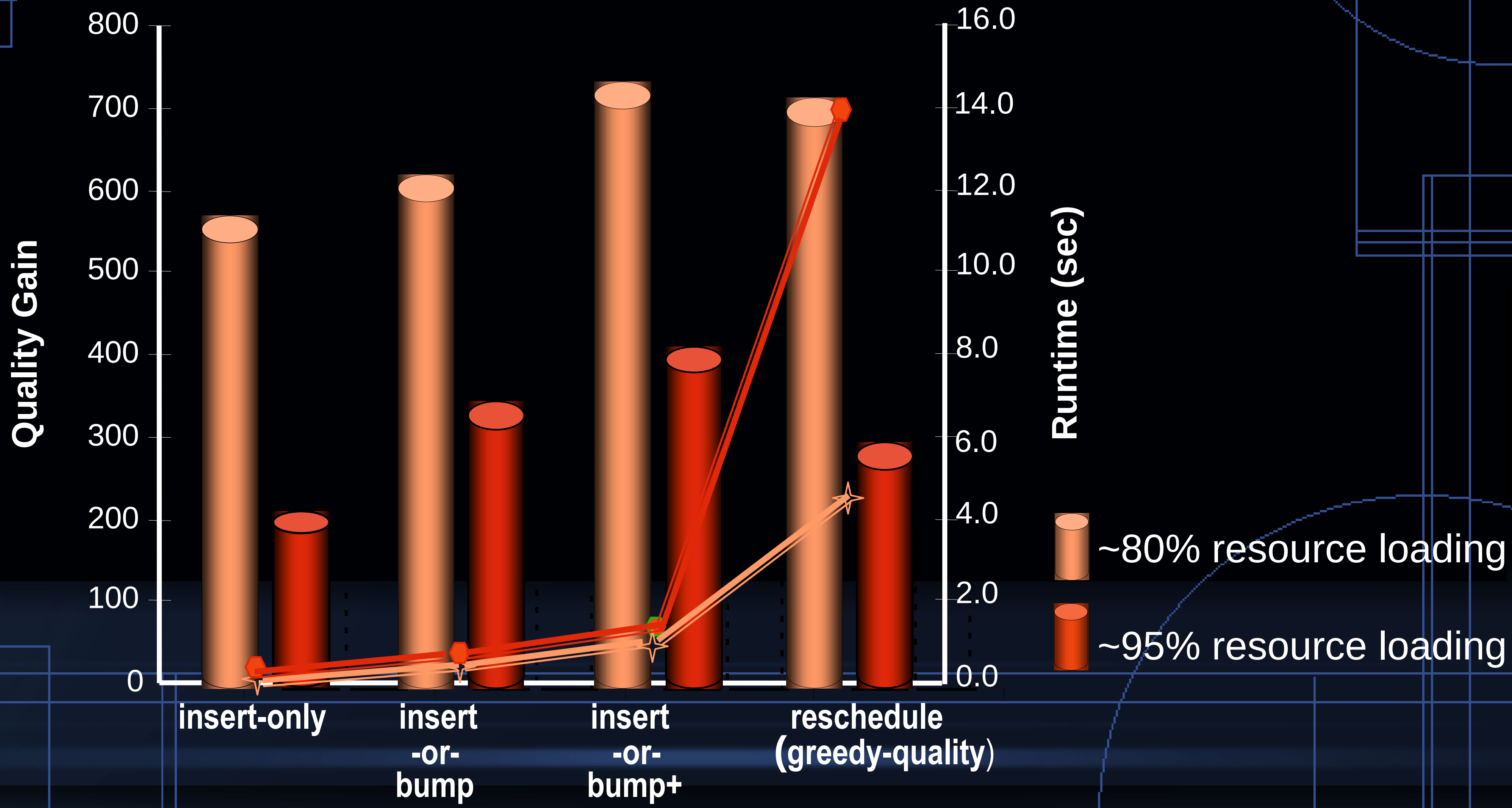
ii. Augment i with option to bump activities already on resource timelines (contiguous sets only)

iii. Augment ii with search for reinsertion of bumped activities (no recursive bumping)

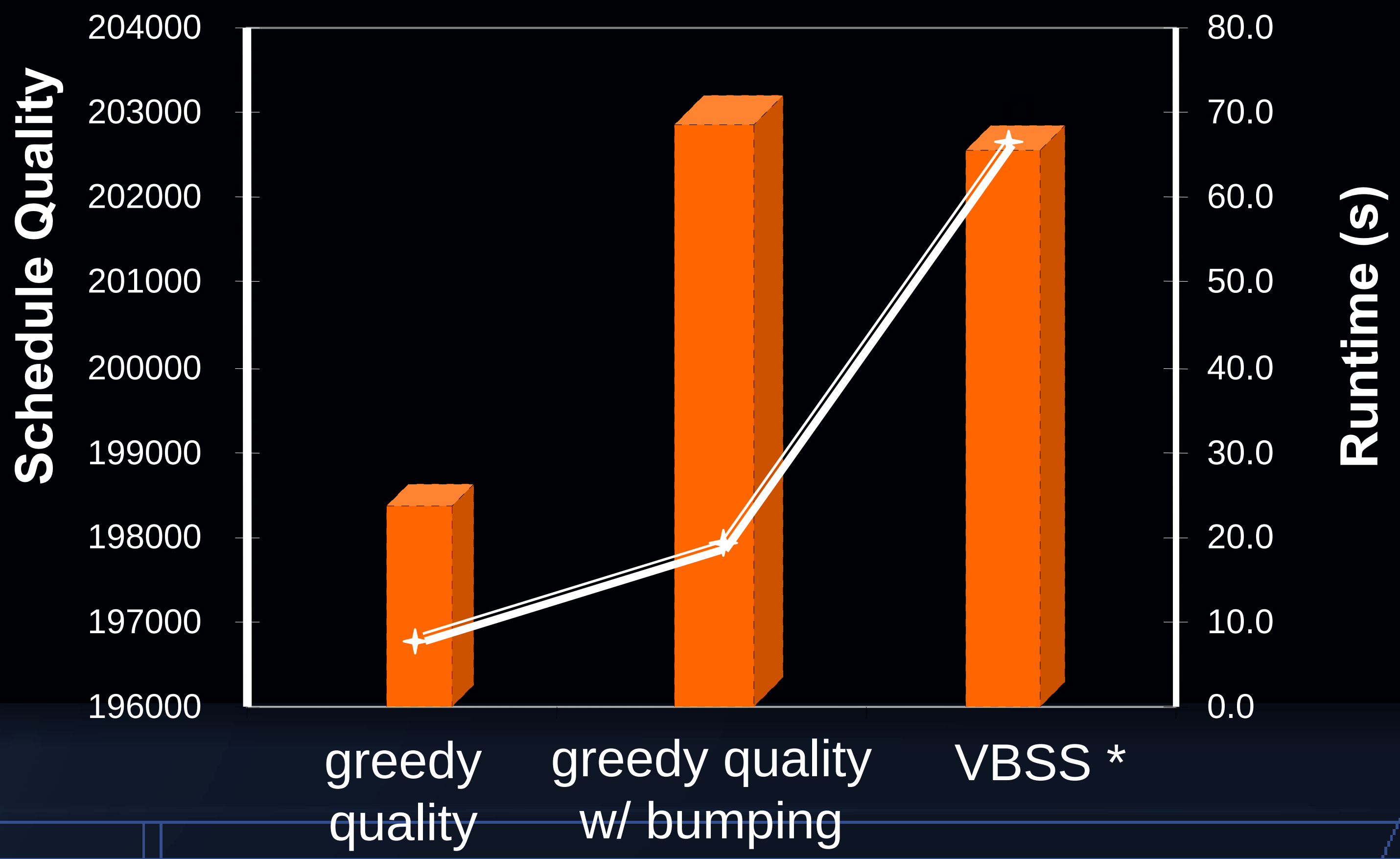
Compared to:

Reschedule from scratch with the newly added activity included in the agenda.

Dynamic Incremental Strategies vs. Rescheduling



Extended *rescheduling* strategies



* heuristic-based stochastic sampling

Preference for more stable schedules

Characterization of Solution Stability

Minimizing schedule perturbation when responding to new events can be key to maintaining execution coherence.

3) $P_{schedule} = w_1 \, x \, b + w_2 \, x \, r + d$

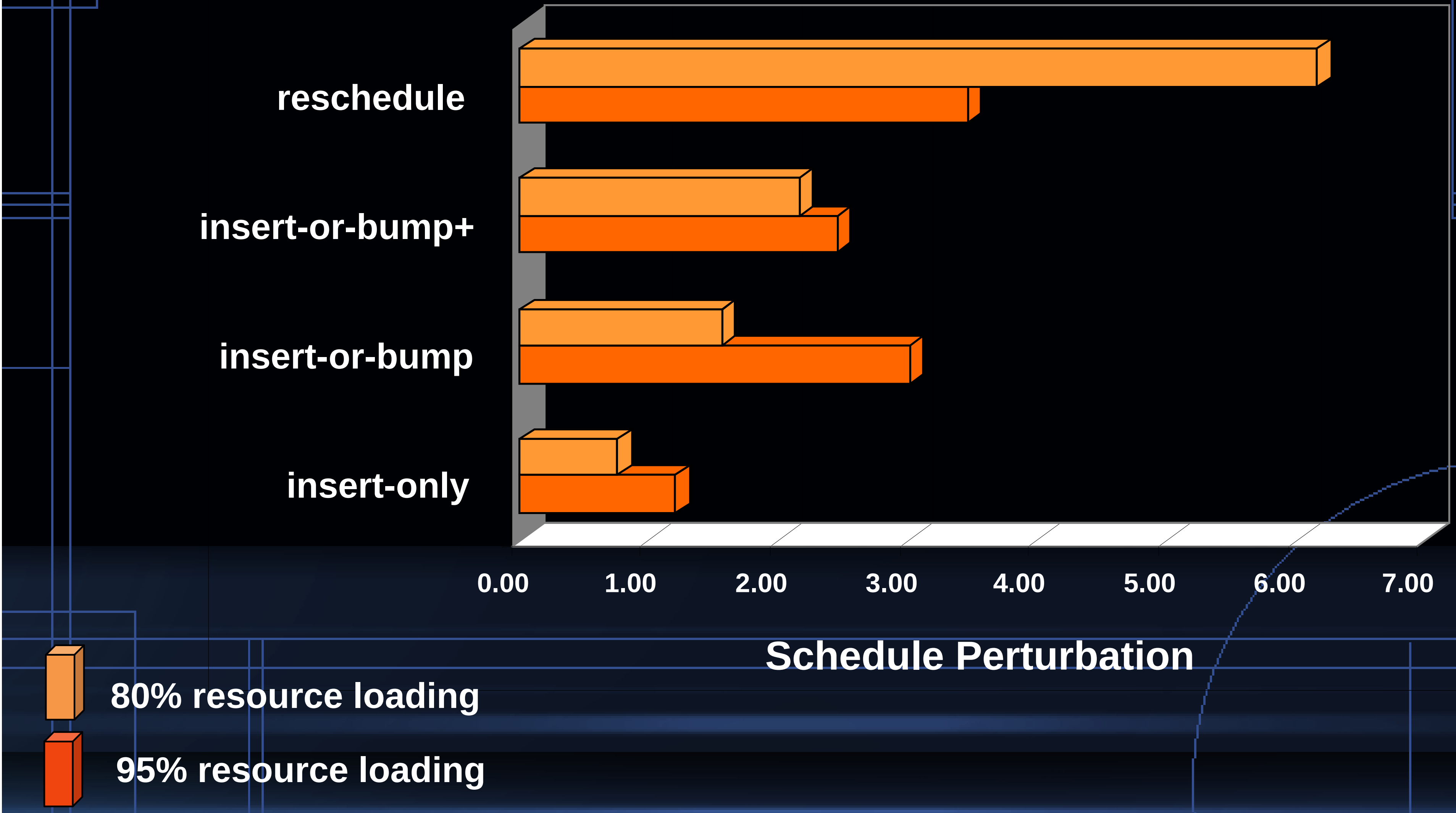
where b: no.of bumped activities

r: no. of activities transferred to a different resource

d: no. of activities with duration changes

w_1 & w_2 : weighting factors (3, 2)

Characterization of Solution Stability



Related Work

- Reactive scheduling (Smith '94; Zweben *et al.* '94; Becker & Smith '00; Sakkout &Wallace '00; Bartak, Muller, & Rudova. '04)
- Trade-off between service time on-task & quality: *IRIS* (Dey, et. al.), *imprecise computation* (Lui, et. Al), *progressive processing* (Mouaddib & Zilberstein '98), *design-to-time scheduling* (Garvey & Lesser, '93)
  - Typically single agent
  - CPU scheduling: pre-emptible, no setup
- Operations Research: time/cost tradeoff problem
  - –Infinite capacity variation of the quality-centric KIDS problem

Generally, none of these studies are concerned with optimizing quality.

Soft constraints and preferences: HTN models of quality accumulation

TAEMS and CTAEMS developed for:

- Collaborative, distributed management of joint plans/ schedules
- Agents must keep pace with real-time (or near real-time) execution
- Dynamics: New activities may extend agent's view at any time, constraints on existing activities may change
- Assumptions:
  - No agent has complete global view
  - Uncertain execution environment
  - Objective is to maximize quality

Soft constraints and preferences:

HTN models of quality accumulation

CTAEMS Problem Representation

- **HTN-style representation of problem constraints**
- **Methods (leafs) are primitive process steps**
  - Executable by a specific designated agent
  - Have associated duration and outcome (quality) distributions
- **Tasks model abstract processes and encapsulate sets of more detailed tasks**
  - QAF type - determines how quality is accumulated
- **Tasks/methods can have start and end time constraints**
- **Tasks/methods can have non-local effects (NLEs)**
  - Hard constraints must be enforced - A *enables* B
  - Soft constraints boost (or degrade) quality - A *facilitates* B

Soft constraints and preferences:

Solution Approaches for TAEMS problems

Centralized scheduling

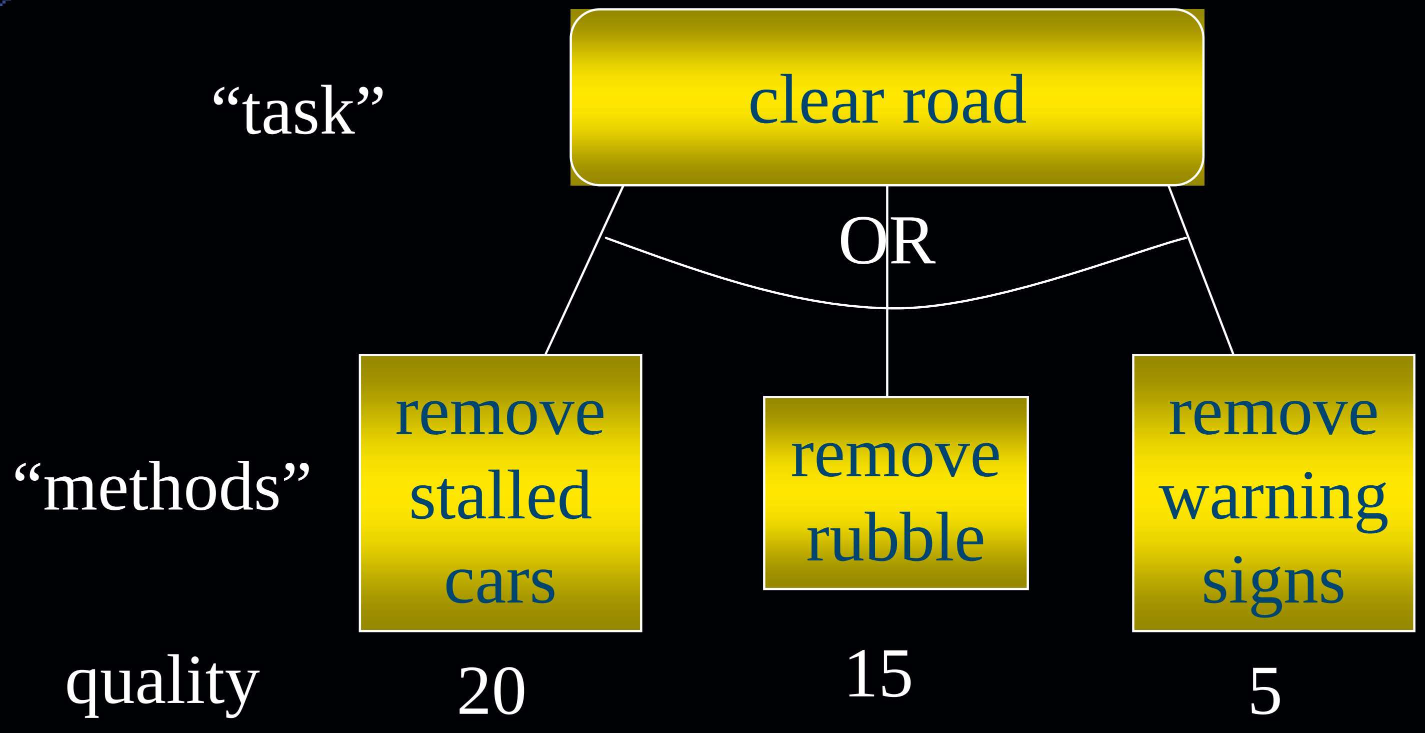
- Criteria Directed Task Scheduling. (Wagner, et. al., 1998, Journal for Approximate Reasoning (Special Issue on Scheduling), Volume 19)
- CP-solver -Constraint programming (Gomes, van Hoeve, and Selman. 2006)
- Incremental constraint-based scheduling with STNs (Barbulescu, Galagher, Rubinstein, Smith, Zimmerman, 2006)

Distributed scheduling

- DTC (Wagner & Lesser, 2000. *Design-to-Criteria Scheduling: Real-Time Agent Control*)
- Constraint programming (van Hoeve, Gomes, Lombardi, and Selman. 2007 *Optimal Multi-Agent Scheduling with Constraint Programming*. (IAAI 2007).
- MDPs (Musliner, et. al. 2006. *Coordinated Plan Management Using Multiagent MDPs*. AAAI Spring Symposium on Distributed Plan and Schedule Management)
- Partial centralization (Szekely, Maheswaran, Neches, Sanchez, 2006. *An Examination of Criticality-Sensitive Approaches to Coordination*. AAAI Spring Symposium on Distributed Plan and Schedule Management)
- Incremental constraint-based scheduling with STNs (Zimmerman, et. al. 2007, *Distributed Management of Flexible Times Schedules* Intl conf on Autonomous Agents and Multiagent Systems (AAMAS).

ctaems task quality accumulation

Sum Task



“task”

clear road

OR

“methods”

remove stalled cars

remove rubble

remove warning signs

quality

20

15

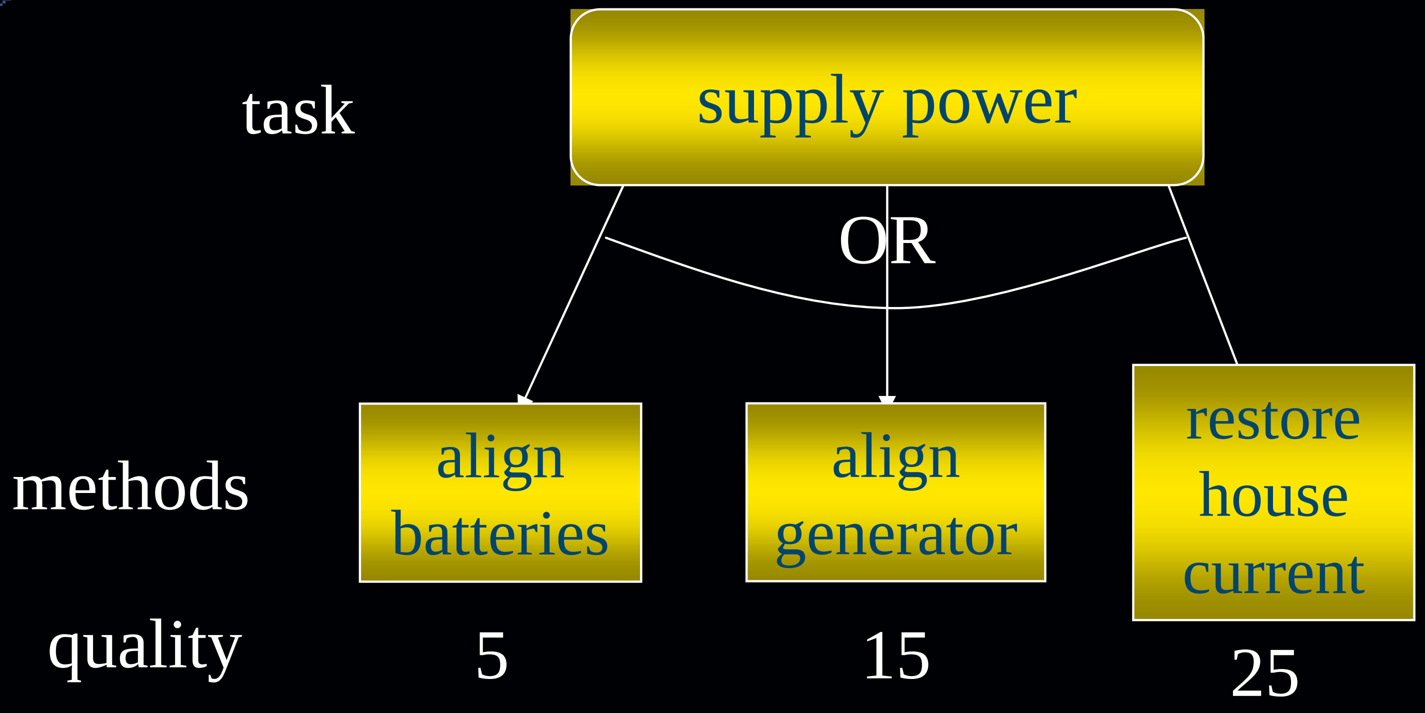
5

Let $S \subseteq Methods$ be the scheduled methods and q_i the quality value for method i. Quality Q accrued for the task is:

$$Q = \sum_{i \in S} q_i$$

ctaems task quality accumulation

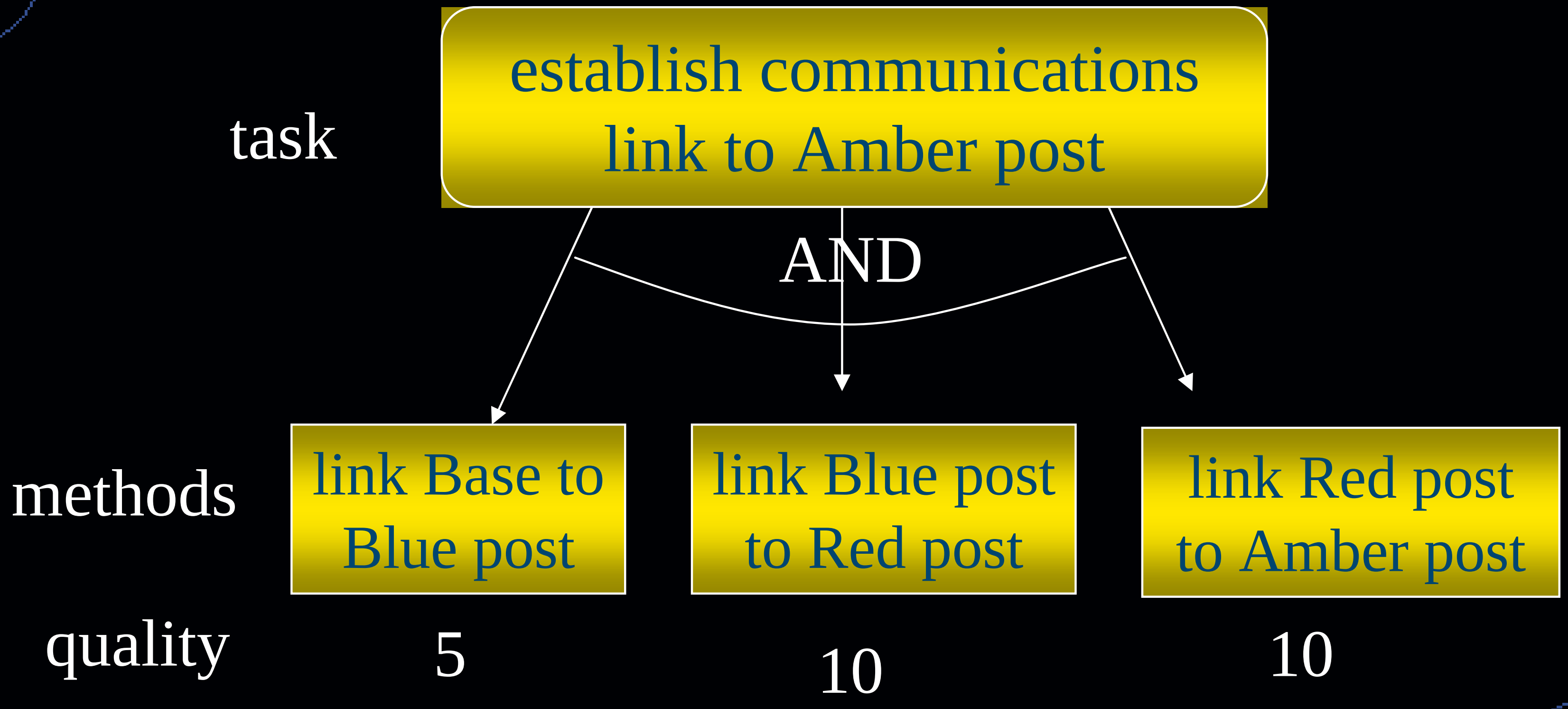
Max Task



Let $S \subseteq Methods$ be the scheduled methods under task and q_i the quality value for method i. Quality Q accrued for the task is: $Q = \max_{i \in S}(q_i)$

ctaems task quality accumulation

Min Task



task

establish communications link to Amber post

AND

methods

link Base to Blue post

link Blue post to Red post

link Red post to Amber post

quality

5

10

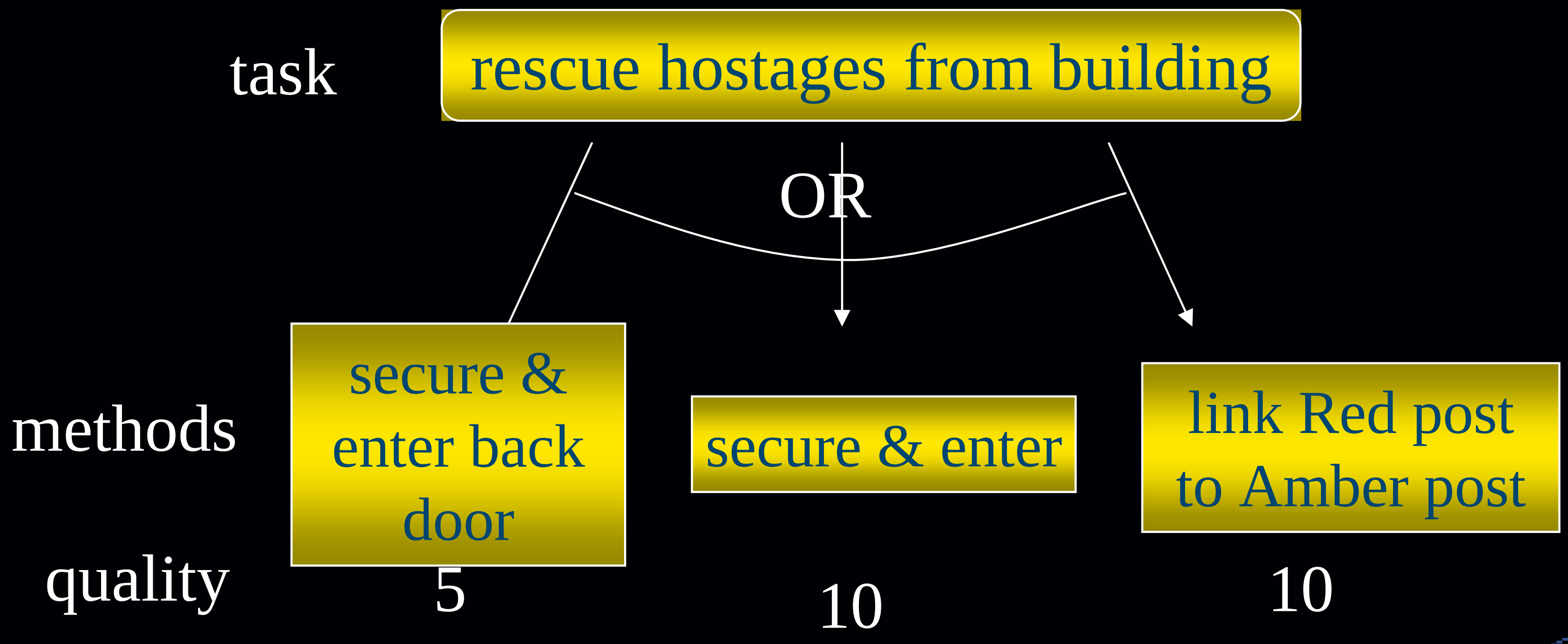
10

Let $S \subseteq Methods$ be the scheduled methods under task and q_i the quality value for method i.

If all methods under Min task are scheduled, quality *Q* for the Min task is: $Q = \min_{i \in S}(q_i)$

ctaems task quality accumulation

SyncSum Task


task

rescue hostages from building

OR

methods

secure & enter back door

secure & enter

link Red post to Amber post

quality

5

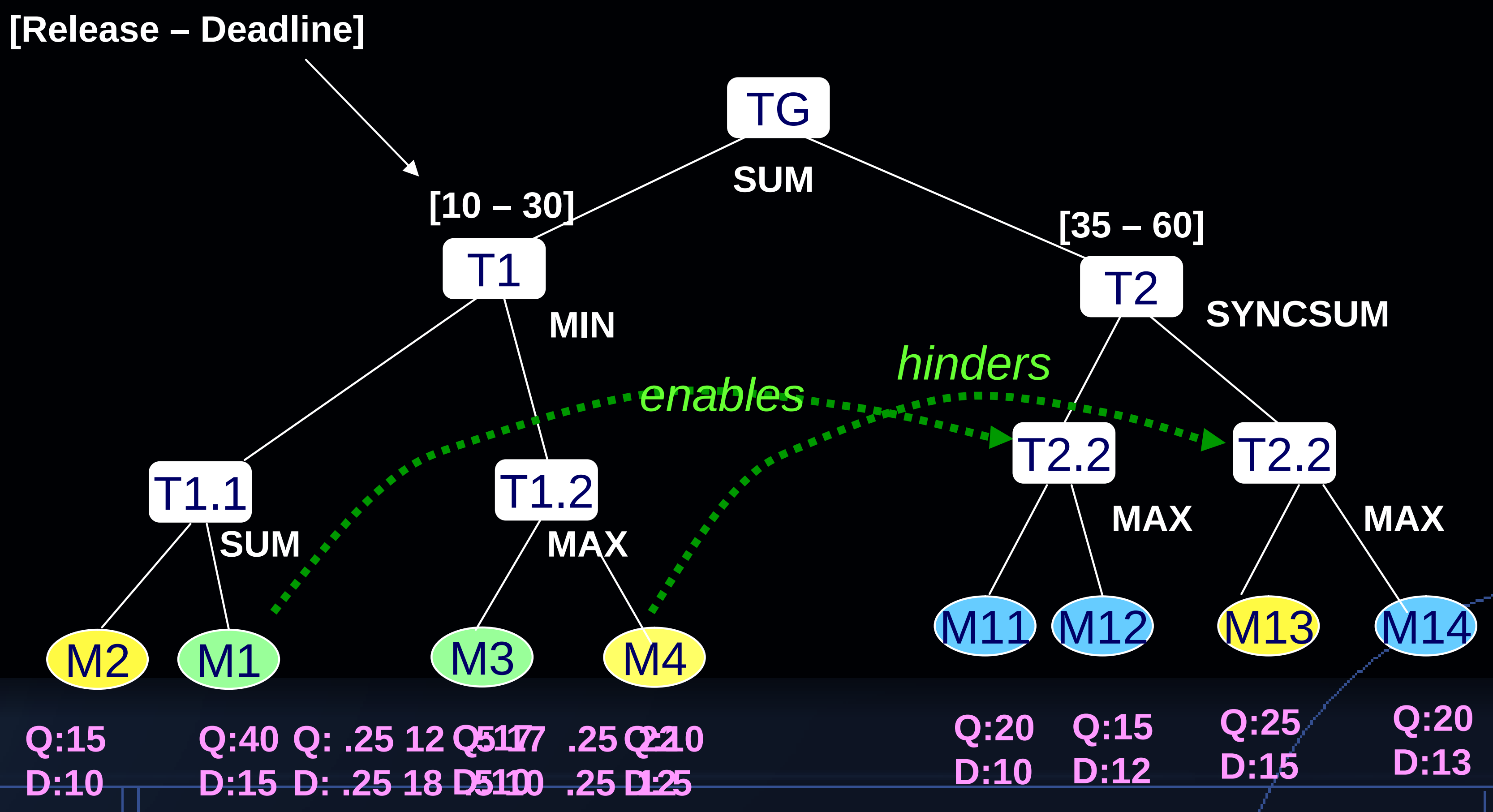
10

10

Let $S \subseteq Methods$ be the scheduled methods and s_1 the first method of S that is executed. Let $A = \{s_i \in S : start(s_i) = start(s_1)\}$ be the subset of scheduled activities that start at the same time as s_1. The quality Q accrued by the task is:

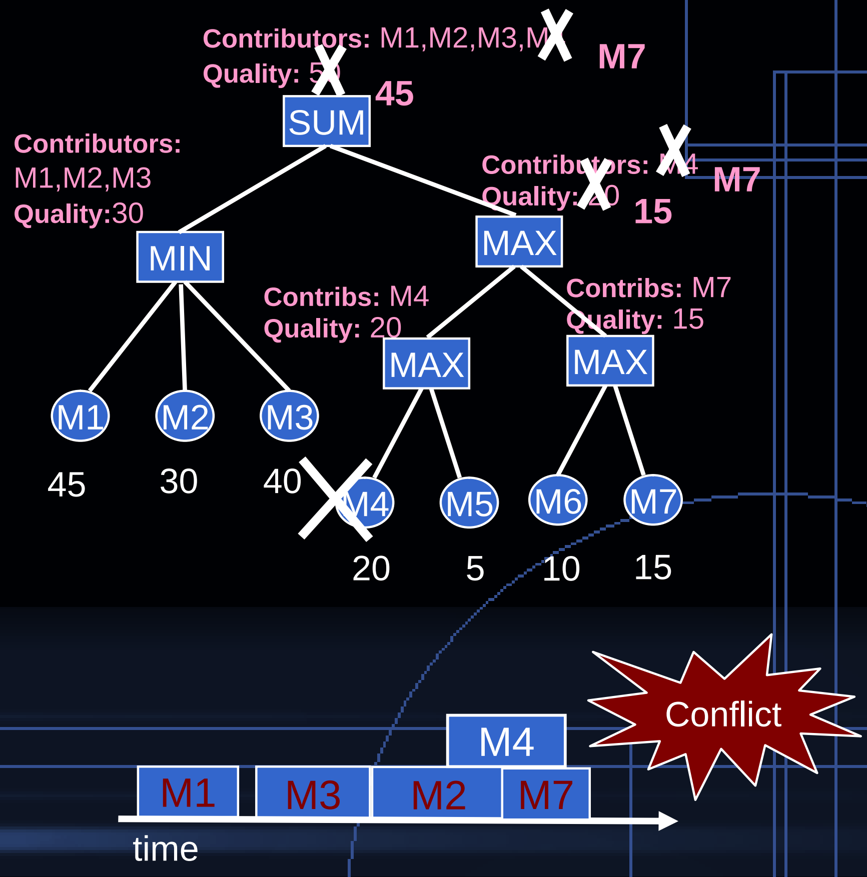
$$Q = \sum_{i \in A} q_i$$

Centralized scheduling over an HTN quality accumulation model

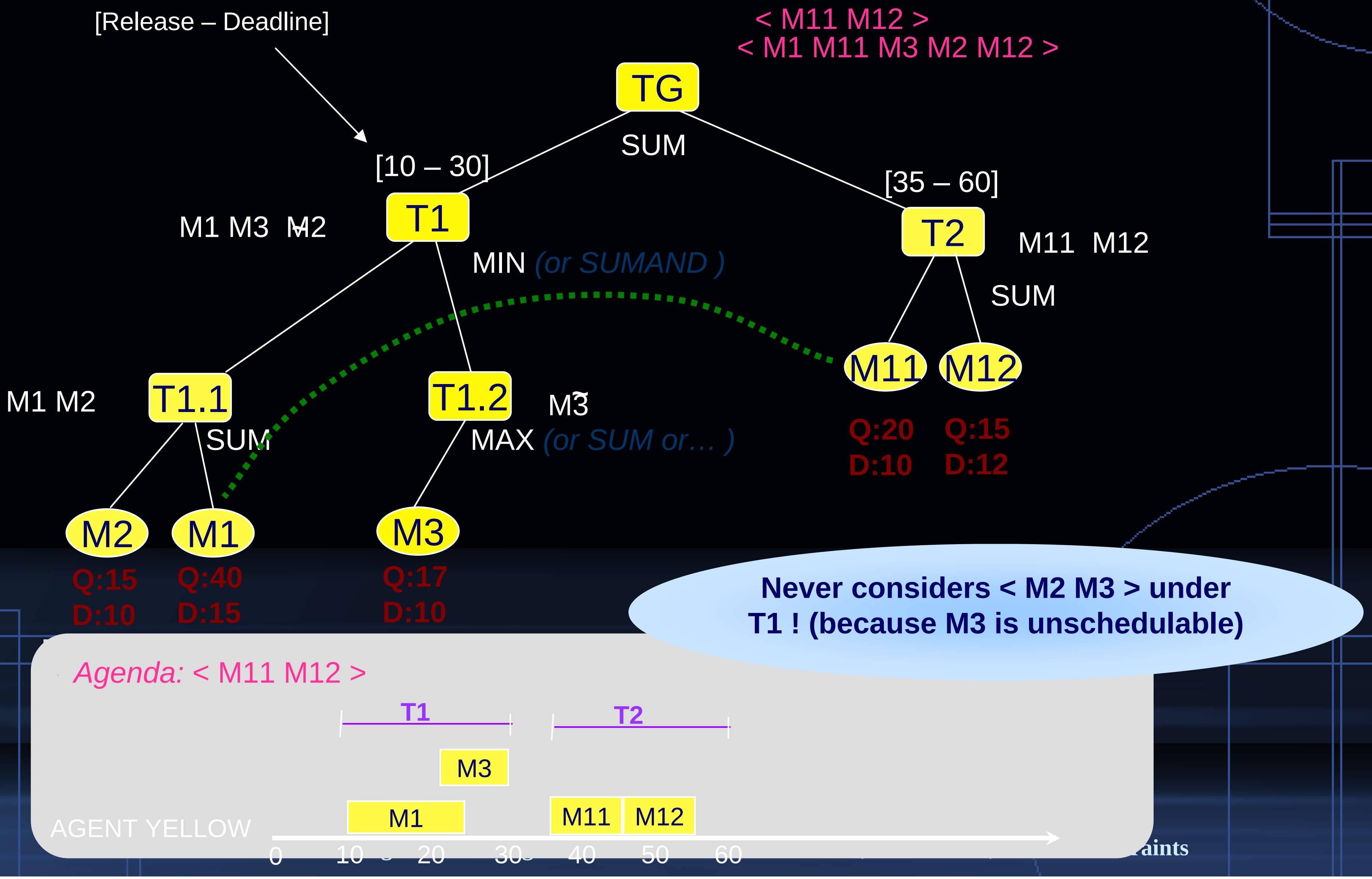


Core Incremental Scheduling Procedure

- Schedule construction/revision via iterative application of two procedures:
    1. ***Quality propagation*** - to compute upper bound on expected (local) quality and identify contributors
    2. ***Method allocation*** - to prioritize and incrementally insert contributor methods (along with supporting enablers) into the agent's schedule
- A scheduling failure at any point retriggers quality propagation, producing a revised set of methods to allocate



Centralized scheduling over an HTN quality accumulation model



[Release – Deadline]

< M11 M12 >

< M1 M11 M3 M2 M12 >

TG

SUM

[10 – 30]

T1

M1 M3 M2

MIN *(or SUMAND)*

[35 – 60]

T2

M11 M12

SUM

M1 M2

T1.1

SUM

T1.2

M3

MAX *(or SUM or…)*

M11 Q:20 D:10

M12 Q:15 D:12

M2 Q:15 D:10

M1 Q:40 D:15

M3 Q:17 D:10

Never considers < M2 M3 > under T1 ! (because M3 is unschedulable)

Agenda: < M11 M12 >

T1

T2

M3

M1

M11

M12

AGENT YELLOW

0 10 20 30 40 50 60

HTN models of quality accumulation:

Incremental Approach vs. Optimal

- 2500 problem instances spread over 5 problem classes
- 3-10 agents, 50-100 methods
- Comparison to expected optimal solution (computed centrally)

Problem Class	% Optimal
OD-AVG	0.979
INT-AVG	1.000
CHAINS-AVG	0.995
TT_AVG	0.949
SYNC-AVG	0.971
NTA-AVG	0.990
OVERALL	0.981

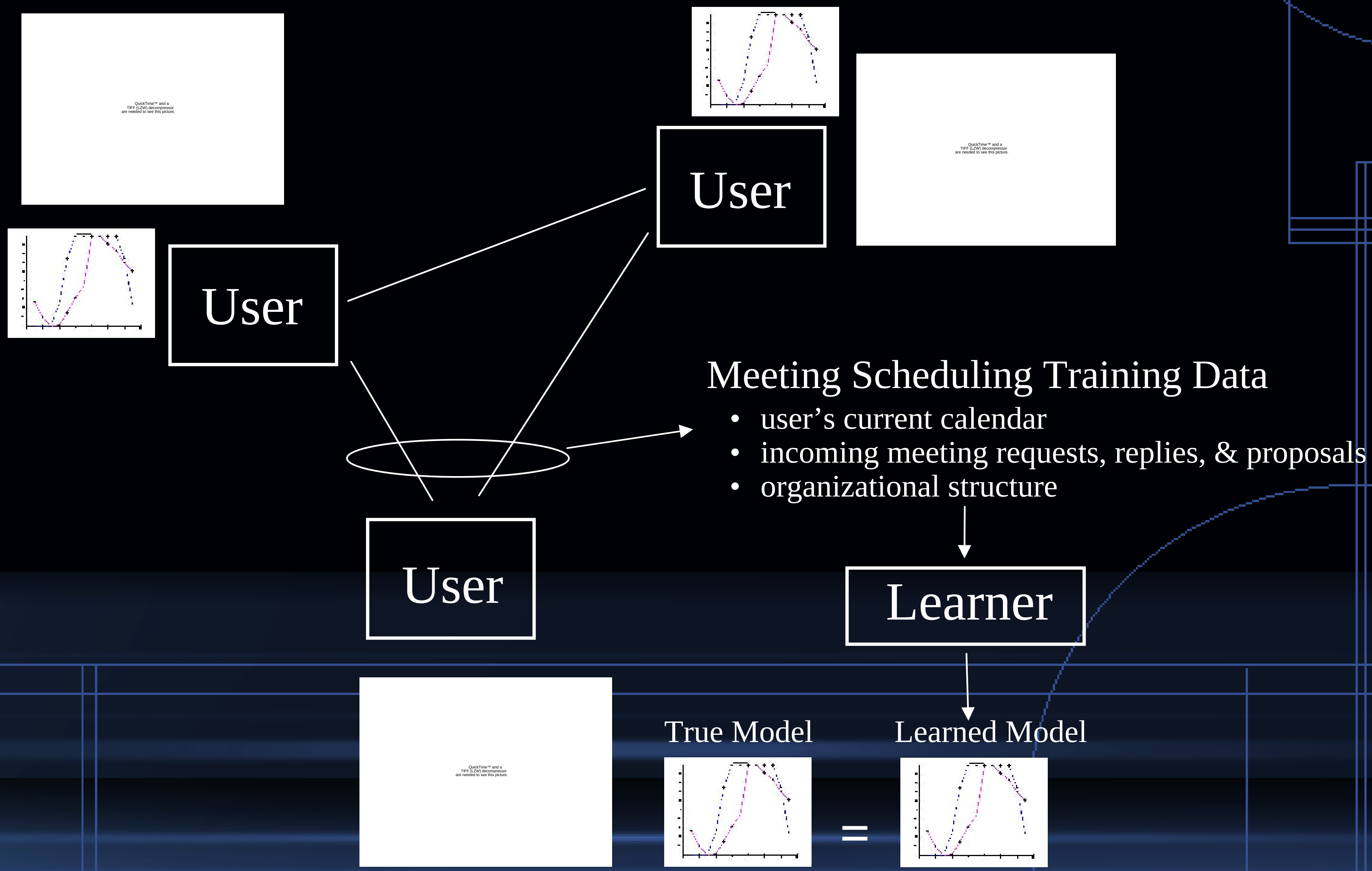
The RADAR Project

(Reflective Agent with Distributed Adaptive Reasoning -a 4-year DARPA sponsored program)

Goal: Create a software personal assistant that

- Understands its user’s tasks, preferences, and changing environment,
- Anticipates and fulfills its user’s information needs.
- Accepts instructions in simple English.
- Handles unexpected situations and requests without reprogramming.
- Improves over time via learning.

Learning Scheduling Preferences



Meeting Scheduling Training Data

- user’s current calendar
- incoming meeting requests, replies, & proposals
- organizational structure

The Changing Planning, Scheduling, and Execution Landscape

- Contemporary sensing technologies (GPS, RF ID tags, ...) now enable collection of real-time information on the location and status of materials and assets
- Technologies for accumulating and managing this information are coming online in various domains
  - Freight and package tracking systems, Manufacturing execution systems (MES), ...

Fine

What’s Missing from the Classical View?

- Practical problems can rarely be formulated as static optimization tasks
  - Ongoing iterative process
  - Situated in a larger problem-solving context
  - Dynamic, unpredictable environment
  - Multiple inter-dependent agents

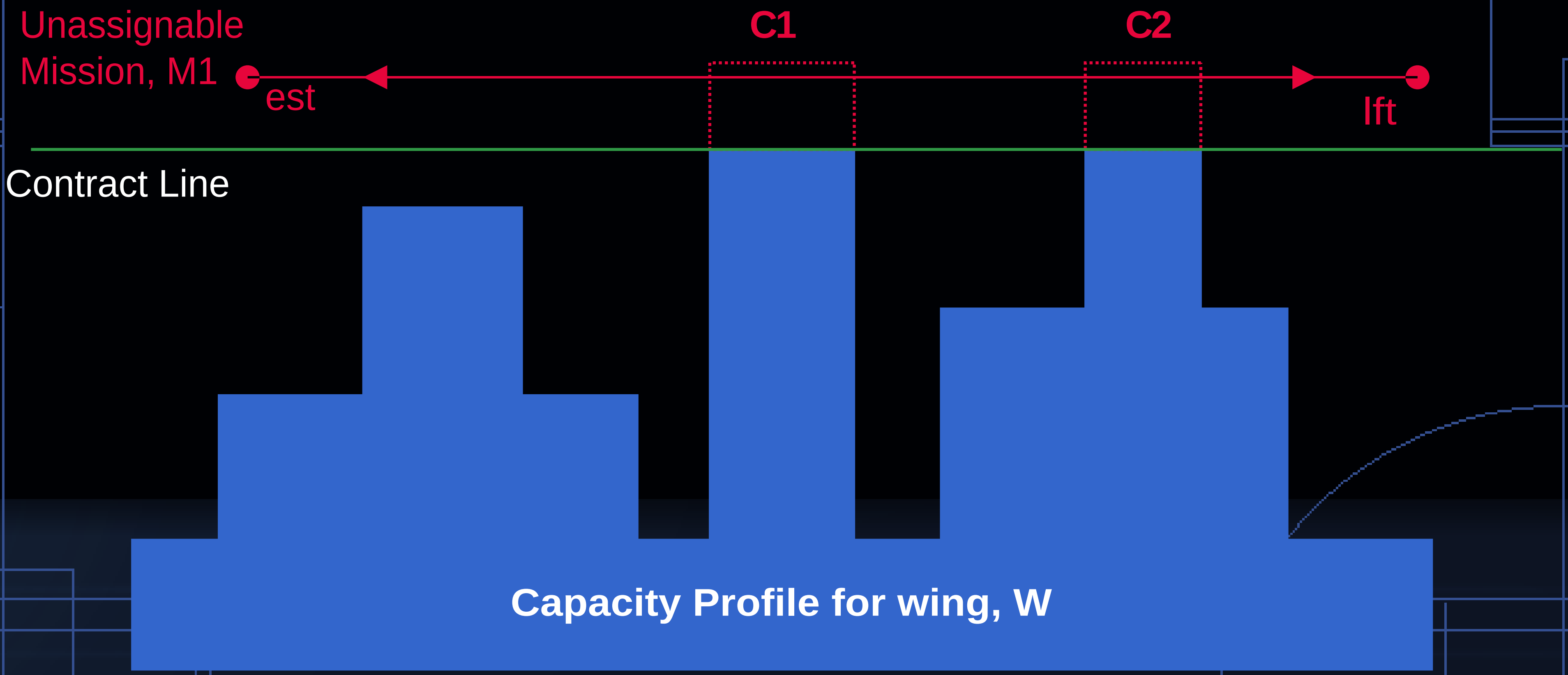
Approaches to Managing Change

- – Build schedules that retain flexibility
- – Produce schedules that promote localized recovery
- – Incremental re-scheduling techniques (e.g., that consider “continuity” as an objective criteria)
- – Self-scheduling control systems

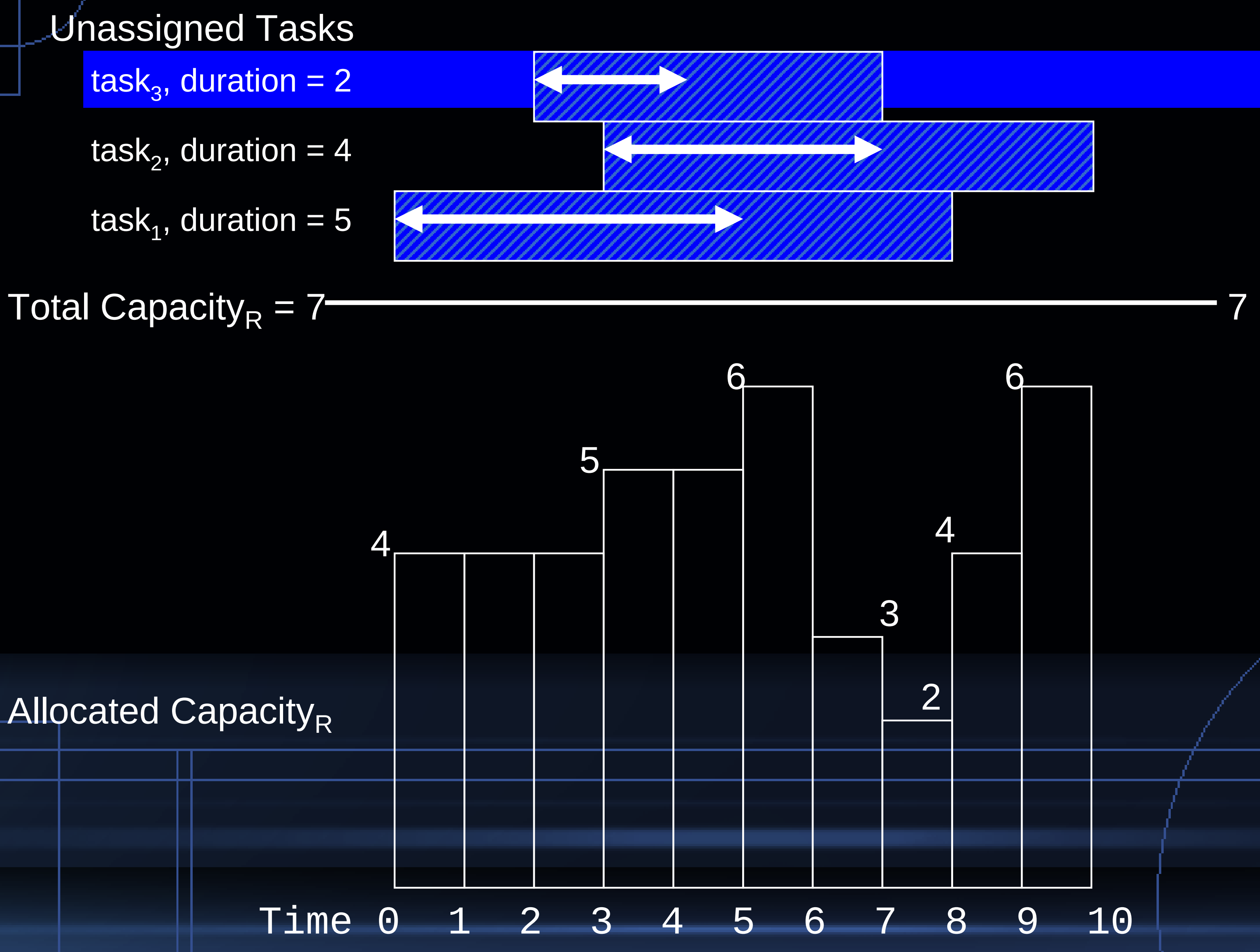
Advantage at Execution Time

- Schedule encapsulates a set of feasible executions
  - – Ability to absorb some amount of temporal delay (or speed up)
- As long as non-empty set of possible start times for each task remains, poly-time re-computation of bounds

Identify where conflicts exist



More recent work: heuristics for task insertion.



Unassigned Tasks

$task_3$, duration = 2

$task_2$, duration = 4

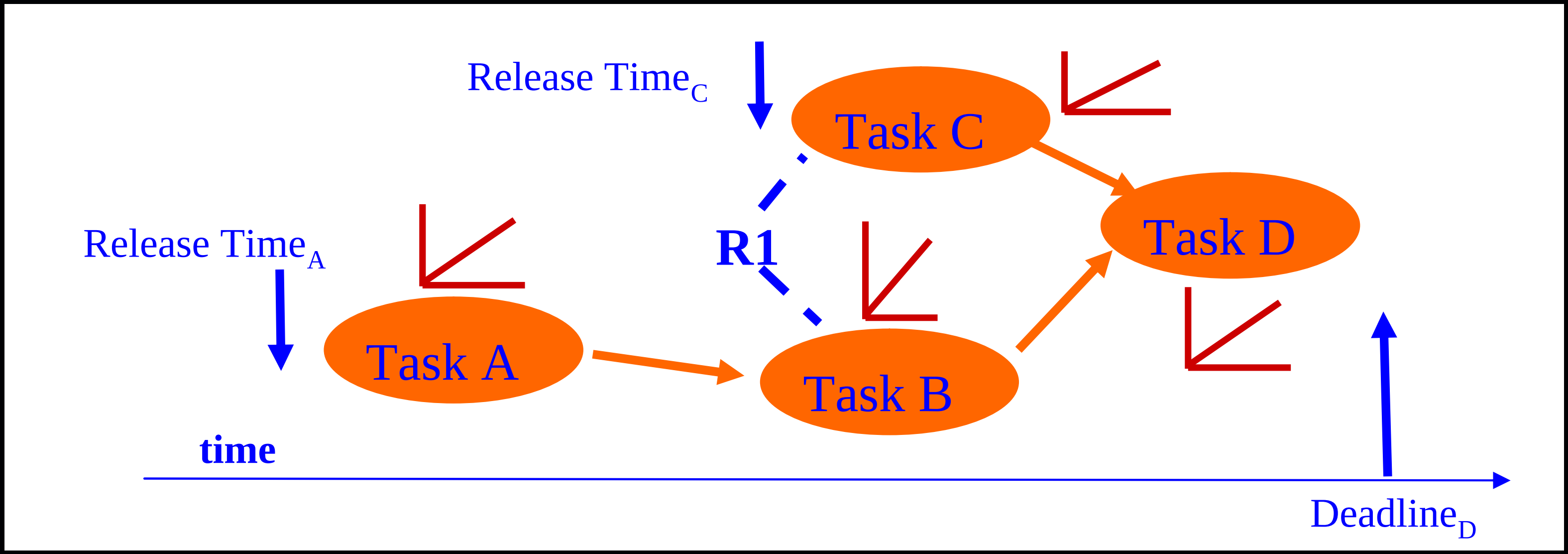
$task_1$, duration = 5

Total $Capacity_R$ = 7

Allocated $Capacity_R$

Hybrid Models for Maximizing Quality

Wang & Smith 2004, 2005

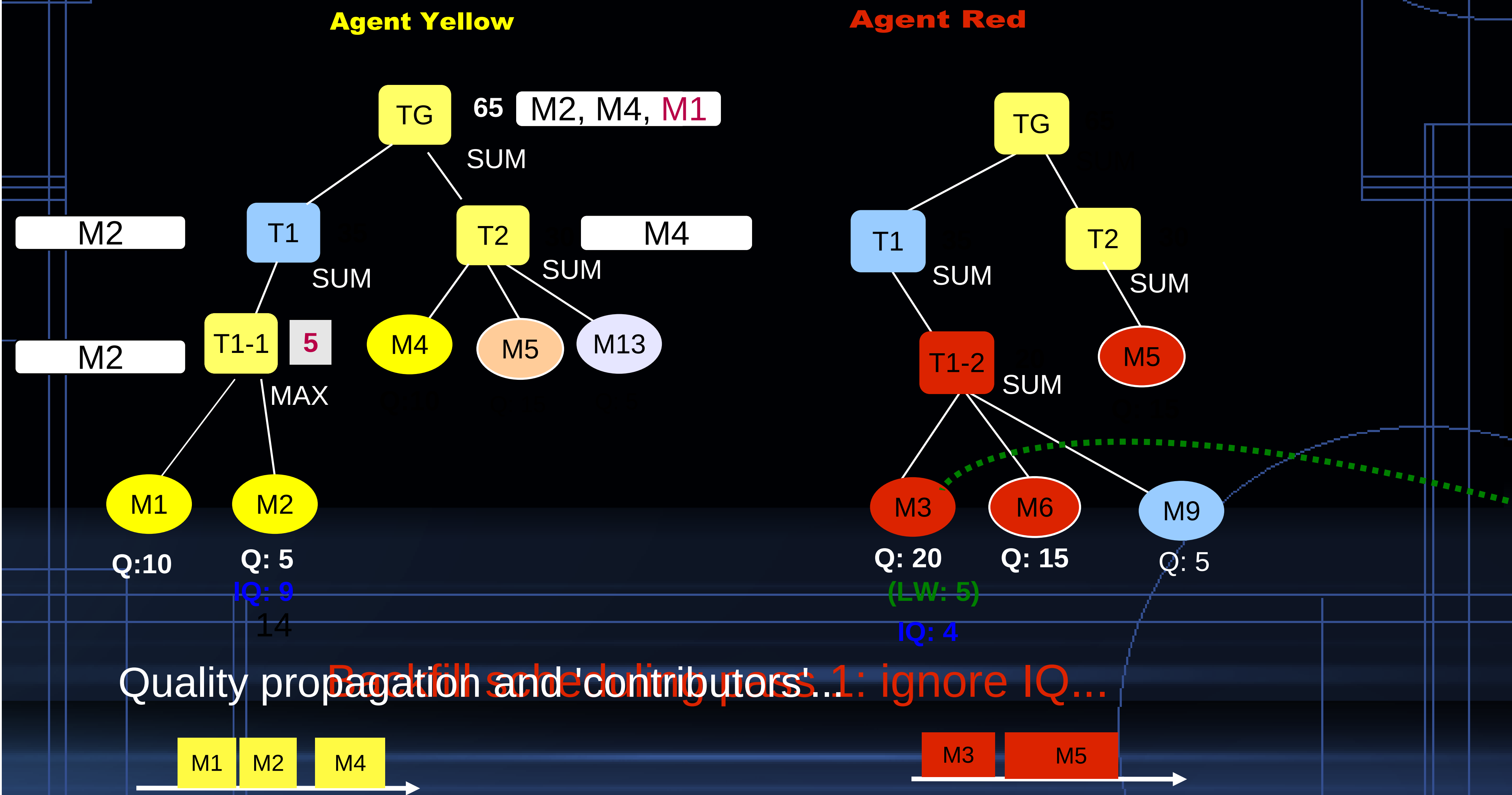


- Observation:
  - – Linear profile, infinite capacity model is efficiently solvable - variation of "linear time-cost tradeoff" problem
- Leads to augmented partial order scheduler

maximizing quality = maximizing fluidity?

Using Impact Quality

SUM / MAX examples for Scheduled Activities



Quality propagation and 'contributors'...

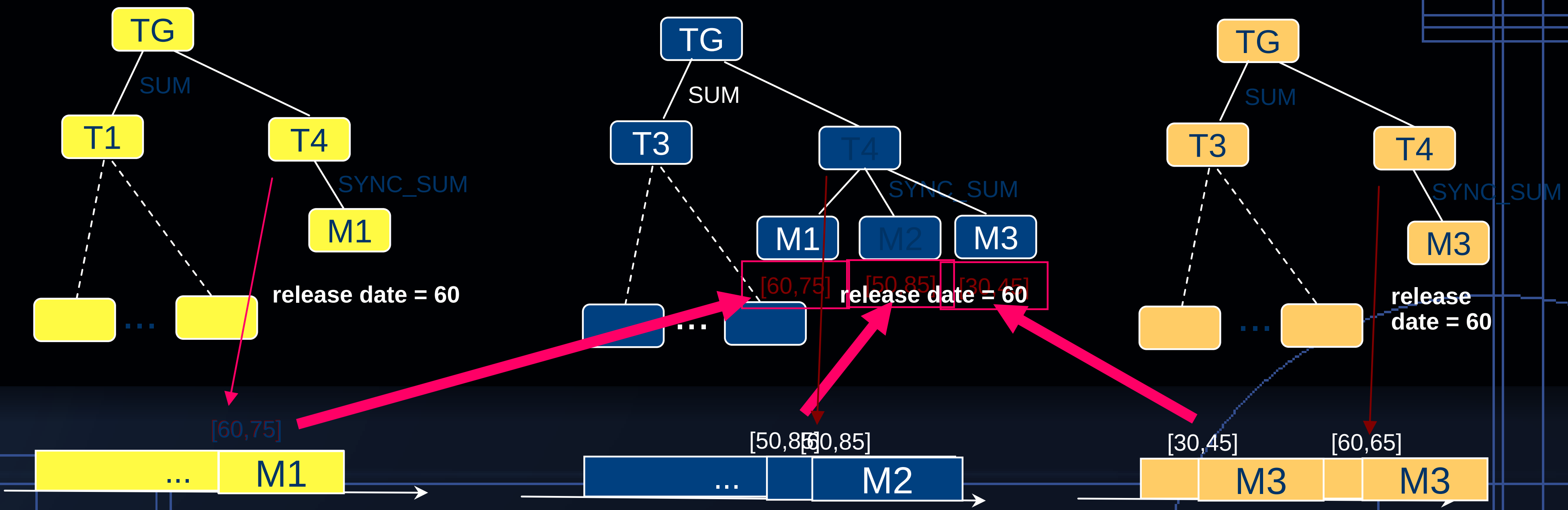
Backfill scheduling pass 1: ignore IQ...

Explicit Coordination over SYNC_SUMs

1. New SYNC_SUM task arrives

2. Agent Blue becomes owner

4. Agent Blue determines maximally inclusive synch point and dictates this new constraint to agents owning task children

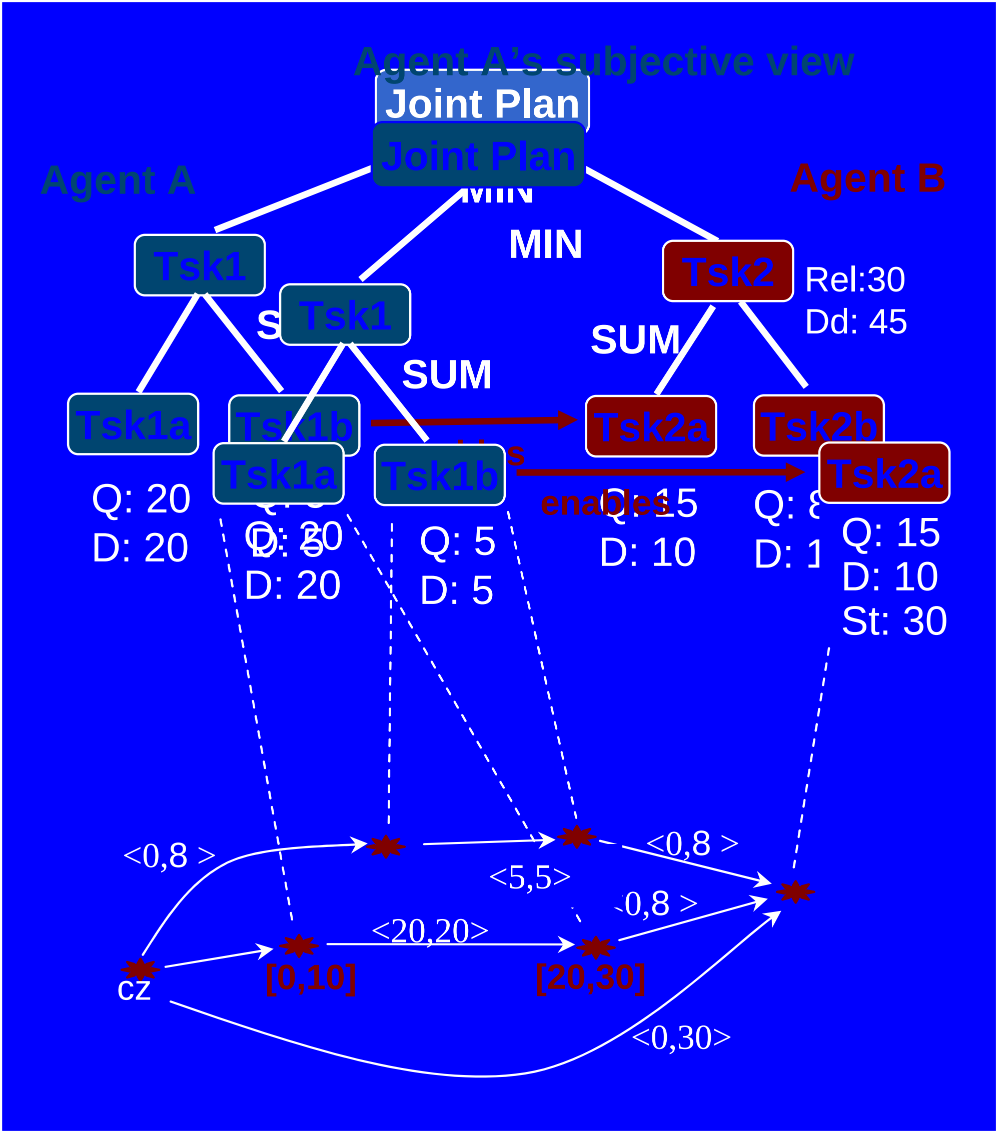


3. Agents schedule locally, and communicate scheduled start time windows to task owner

5. Agent Tan attempts to reschedule M3 to satisfy the synch point

Example

- A given agent only see's its subjective view
- Dual representation within agent maps CTAEMS model to STN
- Values associated with remote nodes treated as additional constraints



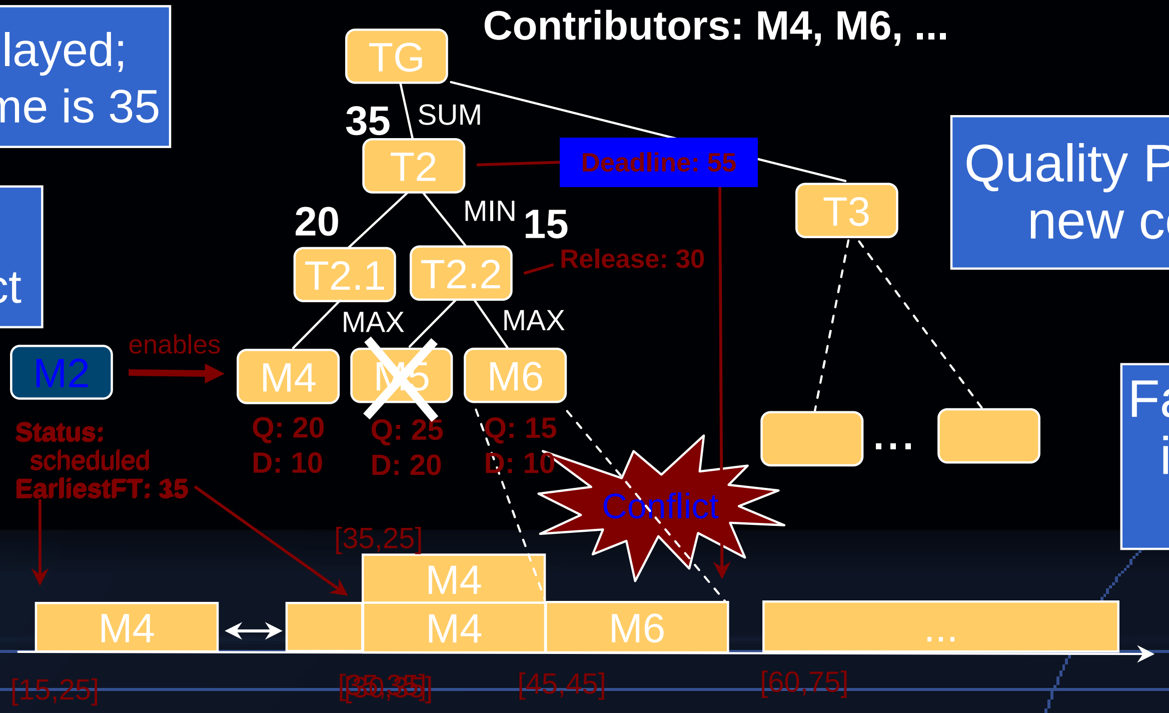
Responding When Execution Dynamics Force Change

- *Conflicts* detected in the underlying STN also signal the need for change and trigger the scheduler

Update: M2 delayed; expected end time is 35

Propagation through STN generates conflict

M5 Retracted to resolve conflict (Reason: M4 necessary for T2 to achieve quality)



Quality Prop indicates new contributors

Fallback M6 is inserted into schedule

Note: A re-solving from scratch approach would unnecessarily re-compute the portion of schedule under T3