Jet Propulsion Laboratory

Automated Planning & Scheduling for Space Mission Operations

Steve Chien, Ari Jonsson, & Russell Knight

Automated Mission Planning

From mission goals

E.g. – image Kilauea (Lat/Lon)

To command sequences

E.g.

```
2003:233:16:49:57 CMD ACSETWHLBIAS(INERTIAL,X=0.341589,Y=1.1749,Z=-0.118046);
2003:233:17:56:57 CMD
  ACGOTOMANEUVER(ORBITAL,TIME=900,XLIMDEG=0.02,YLIMDEG=0.062699,...);
2003:233:18:07:06 CMD I_SETFPEPOWER(POWER_MASK=5);
2003:233:18:07:06 CMD YHEASTBY;
2003:233:18:07:16 CMD YHEASETSWIR(GAINA=1,GAINB=1,GAINC=1,GAIND=1,...);
2003:233:18:07:26 CMD YHEASETVNIR(VNIRALV8,VNIRBLV8,VNIRCLV8,VNIRDLV8);
2003:233:18:11:06 CMD I_CONFIGFPE(CONFIG_COMMAND=16908); ...
2003:233:18:17:06 CMD BCMMODESCRS422;
2003:233:18:17:16 CMD WRMSREC(IDWS=65535,IDWV=65535,...);
2003:233:18:17:54 CMD I_SET_FPE_DG(DURATION=-1);
            ...
```

Why should you be interested in automated planning?

It can offer onboard autonomy to missions

- Enabling response when not in ground contact (earth orbit) or faster response (round trip light for deep space, or rapid automated response)
- It can reduce mission operations costs
- Automation can increase mission reliability in some cases
- Can enable improved mission analysis (more what if cases?)
- …

Outline

Examples
- Sensorweb
- Autonomous Sciencecraft Experiment on EO-1
- Mars Exploration Rover

Committed Planning
- Constraints: State, Resource
- Iterative Repair
- Continuous Planning
- Optimization

Constraint-propagation Planning
- Constraints, timelines, tokens
- Constraint propagation
- Search

Advanced Topics
- Generalized Timelines
- Operations Research Approaches

Practical Issues
- Systems Engineering
- Knowledge Acquisition
- Model Validation, Testing

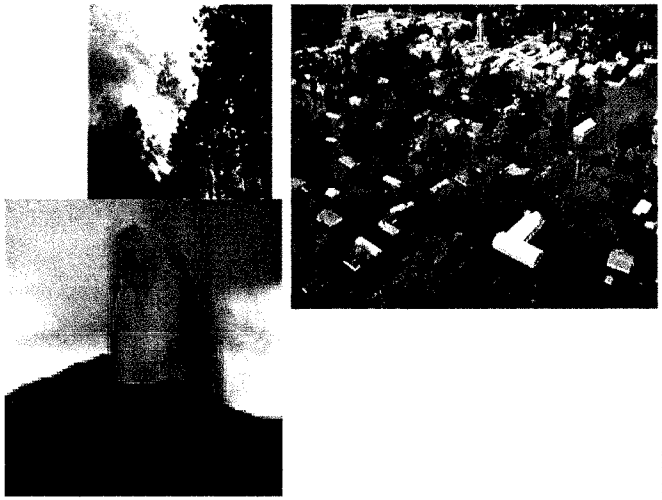
Future Missions enabled by onboard Autonomy

Machine Learning for adaptive autonomy

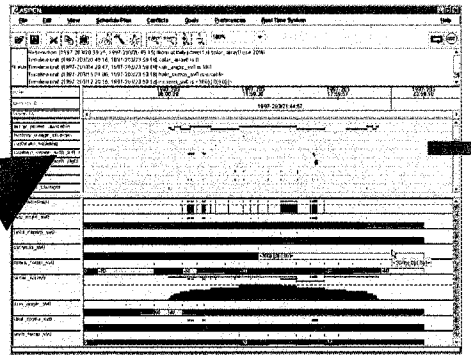
Examples

- Sensorweb
- Autonomous Sciencecraft Experiment on EO-1
- Mars Exploration Rover

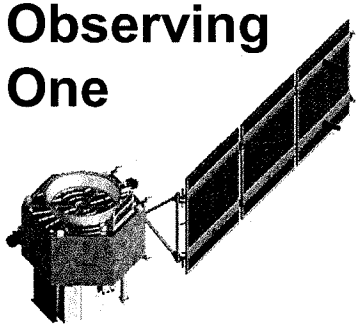
Sensorweb

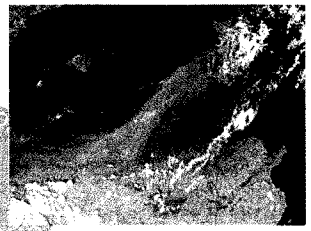


Re-tasking



Earth Observing One



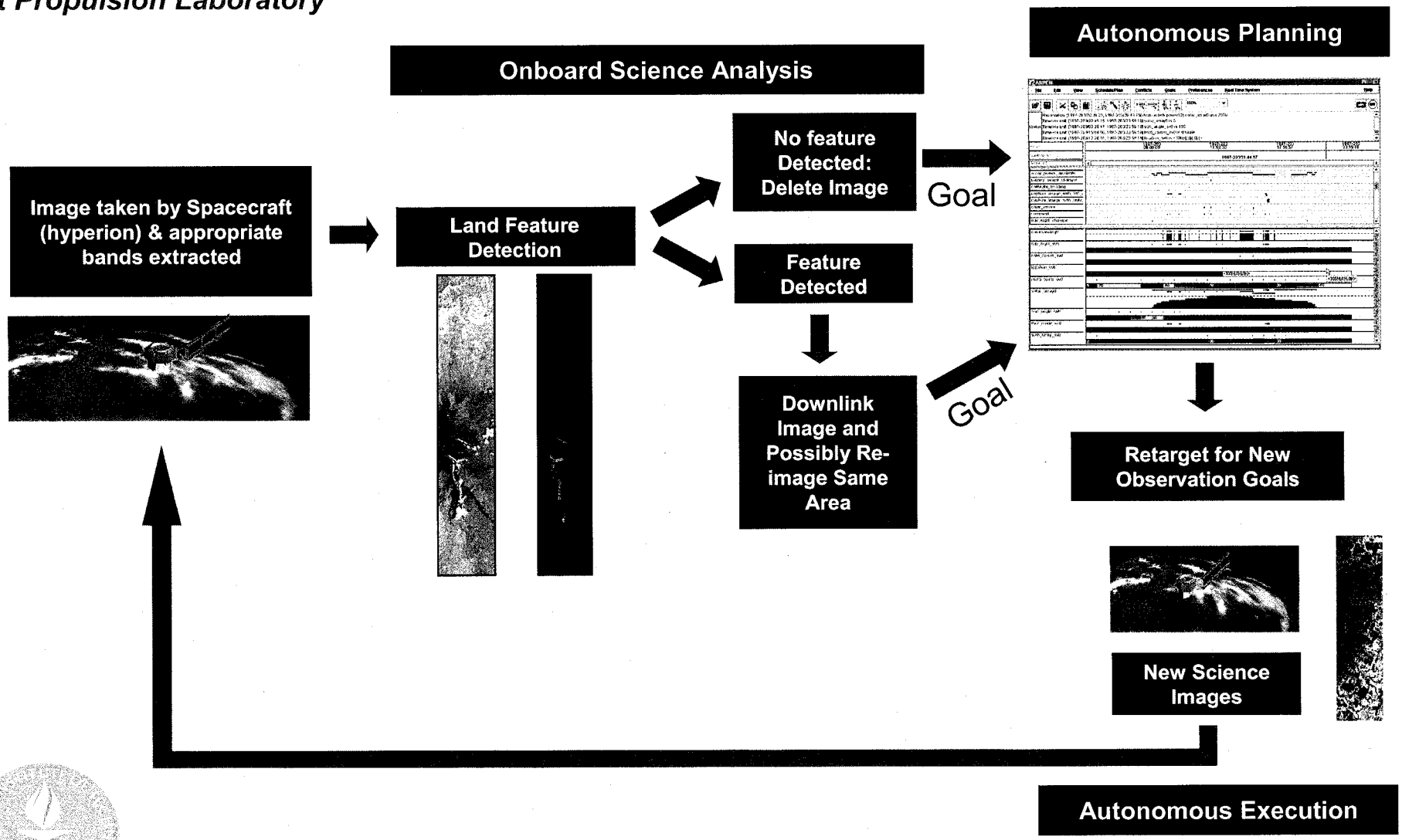


Triggers so far: Wildfires, Floods, Volcanoes (thermal, ash), Ice/Snow, In-situ sensors, modified by cloud cover

Autonomous Sciencecraft Experiment

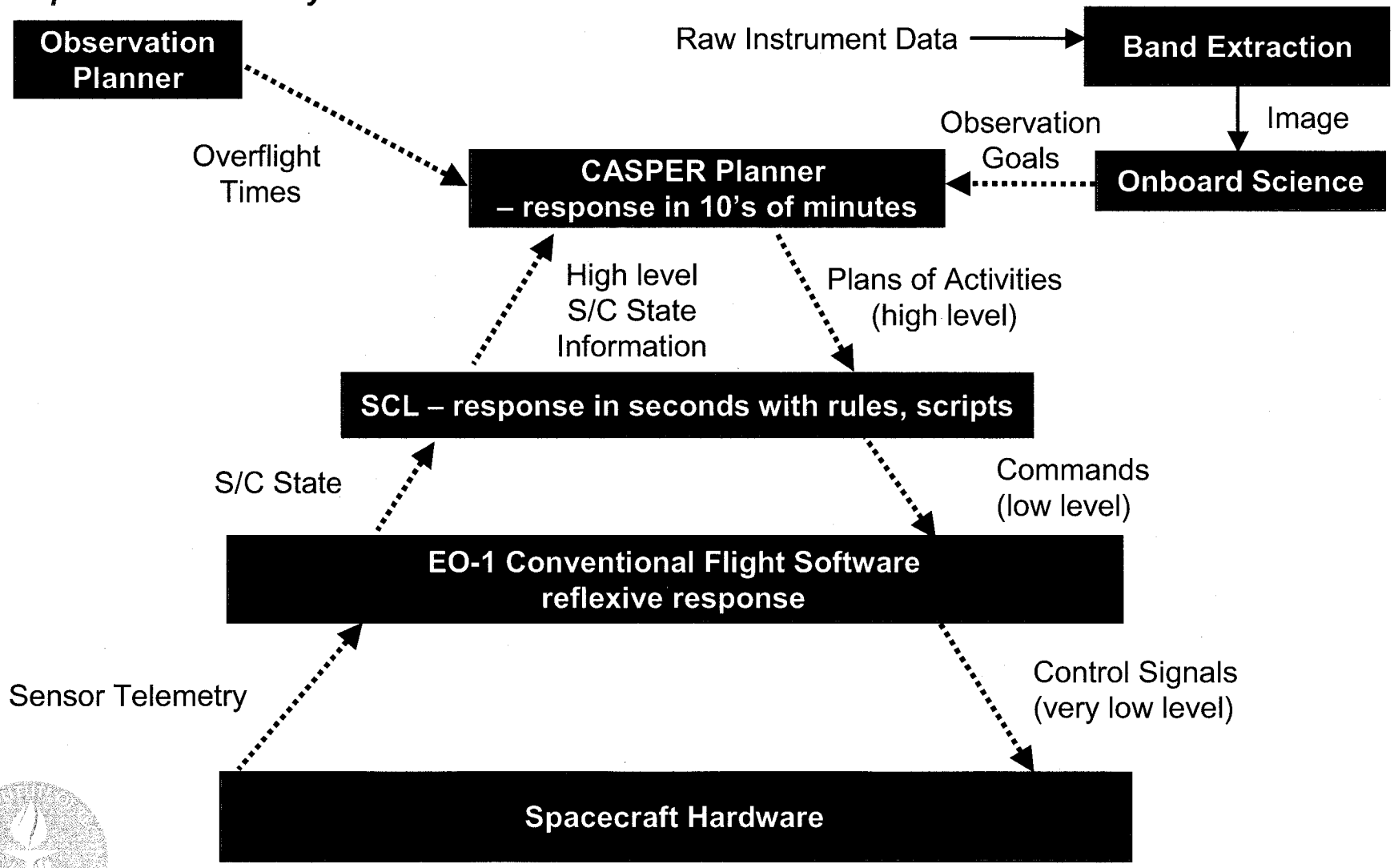
ASE Example Mission Scenario

Jet Propulsion Laboratory



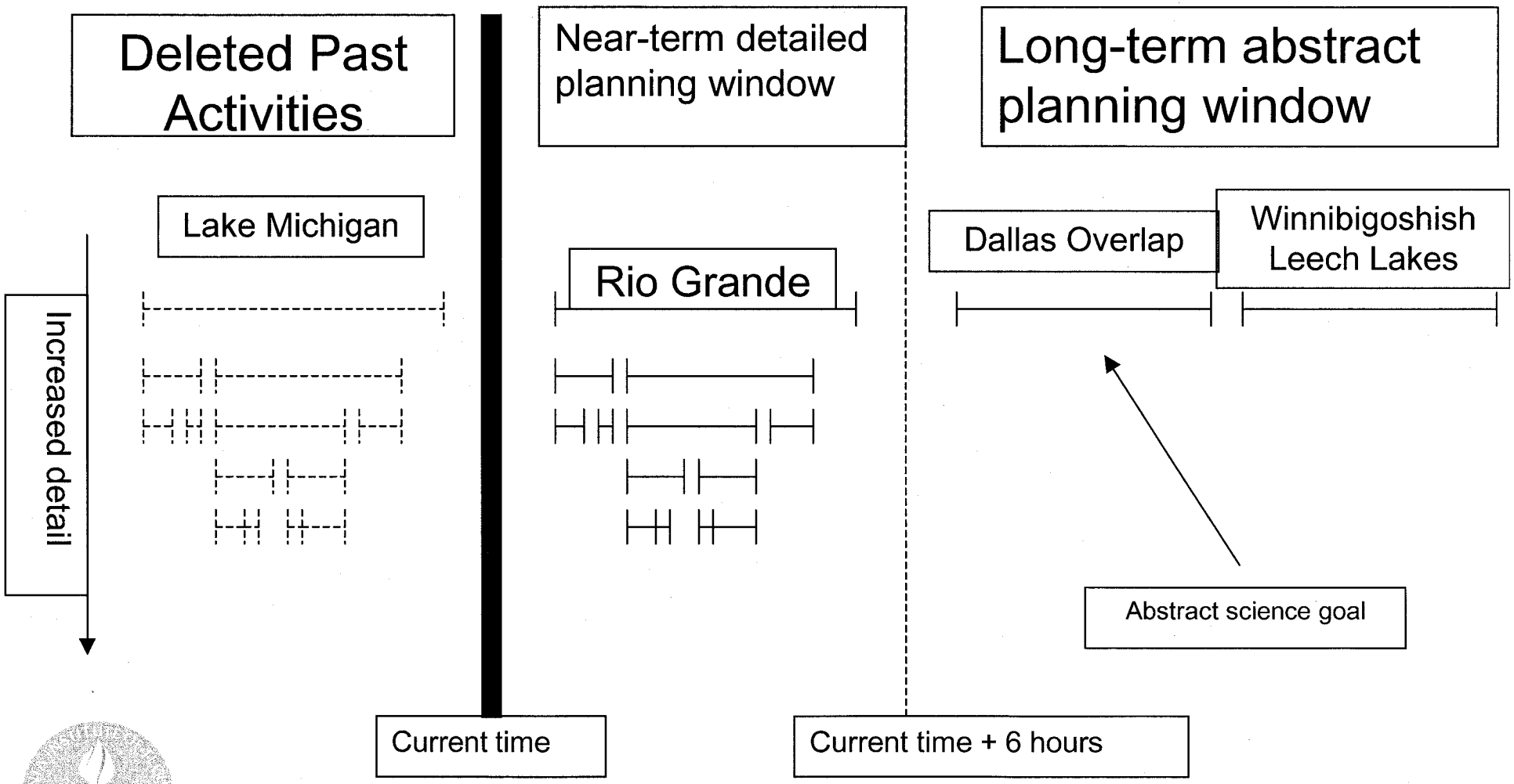
Flight Software Architecture

Jet Propulsion Laboratory



Continuous Planning on ASE

Jet Propulsion Laboratory



Impact of Continuous Planning on EO-1

1 week EO-1 ops = ~100 science observations
+ 50 S-Band/X-Band contacts

= 7800 activities*

= ~ 224 MB Heap Space

CASPER is limited to a 32 MB of heap space

ASE performs detailed planning ~6 hours in advance

= ~16MB Heap Space

* - just for observations, not including the downlink and momentum management activities

Constructive versus Iterative Scheduling Approaches

Constructive approaches

- As decisions are taken, the set of possible schedules is reduced
- Once all decision variables have been assigned, or once a certain condition has been achieved wrt remaining possible schedules (no possibility of resource conflicts)
- Examples: most of the constraint-based techniques we have considered thus far

Iterative approaches

- Search proceeds alternatively by transforming one schedule (or set of schedules) into another
- Schedules are iteratively improved until some termination criteria is met
- Examples: the techniques we are about to consider

Committed Planning

Models

- Describe the environment and all agents in the environment.
- More principled and inspectable than rules

Algorithms

- Solution constructors
- Repair
- Packing

Models

Constraint based modeling system

Activities

- Start time and end time
- Other activities
- Effects on states
- Parameters

Timelines

- Represent the state of the world
- States and resources
- Black-box timelines (Generalized Timelines)

Activity Example

Warmup Activity

- Can be assigned a warmup rate and a destination temperature.
- Consists of a heaterOn activity and a heaterOff activity.
- Its duration is a function of the starting temperature and the destination temperature.
- Its power use is a function of the warmup rate.
- Its energy use is a function of the warmup rate and the duration.

State Example

Filter wheel

- States are red, green, blue, closed
- Wheel turns only clockwise, so allowed transitions are from red to green, green to blue, blue to closed, and closed to red.

Resource Example

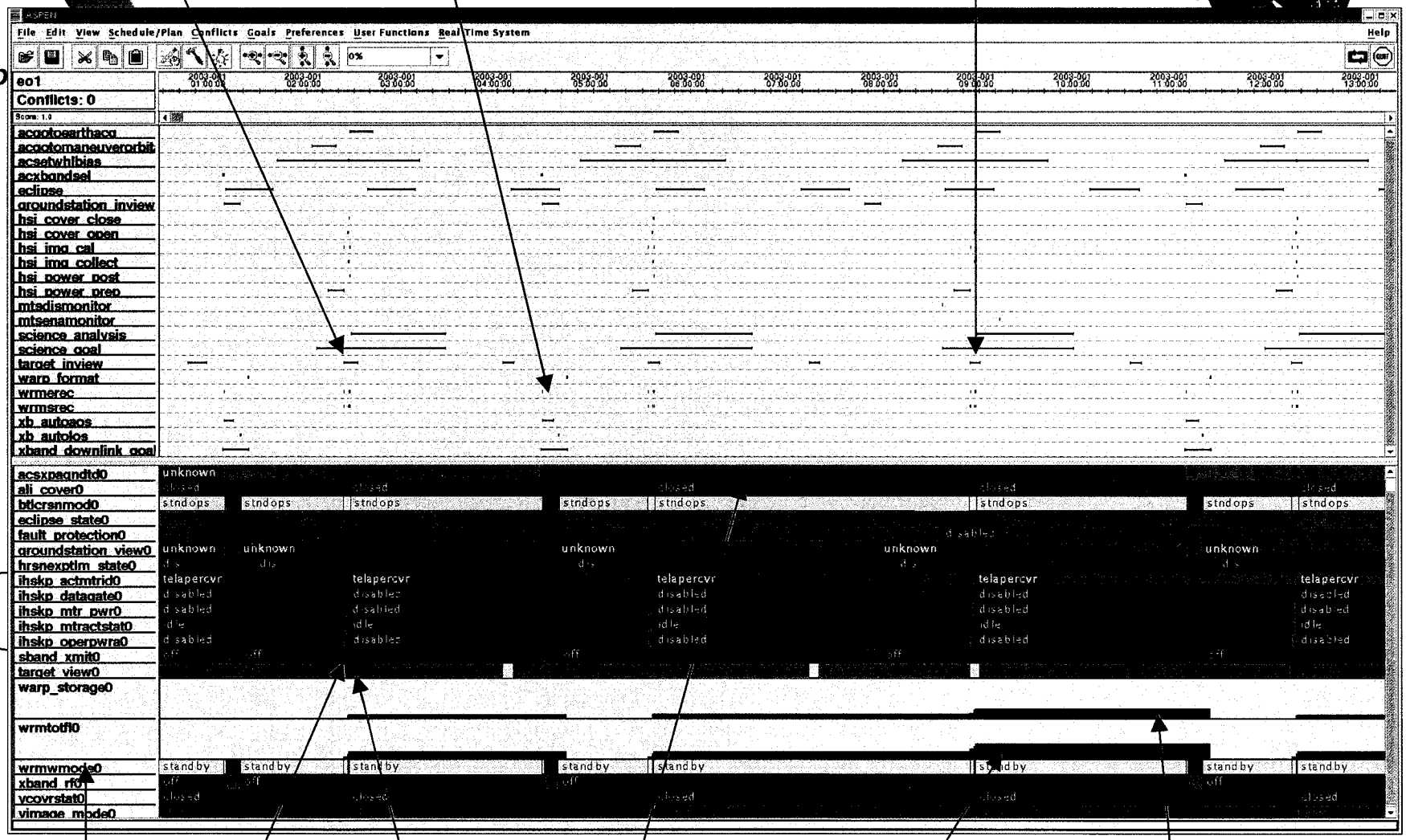
Power

- Non-persistent (non-depletable)
- Minimum power level must be 10
- Maximum power level is 100

Energy

- Persistent (depletable)
- Minimum energy level is 0
- Maximum energy level is 100

Jet Prop


Day collect
Downlink
Night collect
File Edit View Schedule/Plan Conflicts Goals Preferences User Functions Real Time System
Help
eo1
Conflicts: 0
acsxpagndtd0
ali_cover0
bticrsnmod0
eclipse_state0
fault_protection0
groundstation_view0
hrsnexptim_state0
ihskp_actmtrid0
ihskp_datagate0
ihskp_mtr_pwr0
ihskp_mtractstat0
ihskp_operpwra0
sband_xmit0
target_view0
warp_storage0
wrmtotfl0
wrmwmode0
xband_rf0
ycovrstat0
vimage_mode0
Hyp State
Warp mode
Hyperion Preparation
Target in view
X-band Ground Station
WARP file count
WARP data volume

Algorithms

Solution constructors

- Forward Dispatch (Greedy – in order of time)

Repair

- Conflicts
- Repair Methods
- Choice Points
- Heuristics

Packing

- Earliest
- Latest
- Loosest

Forward Dispatch

Narrow the times for activities according to know temporal constraints and everything that has been scheduled so far.

Choose an activity to schedule based on “earliest deadline” heuristic.

Schedule the activity while trying to minimize constraint violations (may not be in order)

Repeat until all activities are scheduled.

Iterative Repair

- Identify the conflicts
- Choose a conflict to repair
- Choose a method to repair the conflict
- Based on the method and conflict, make any other associated choices
- Attempt the repair based on the choices
- Repeat until no more conflicts exist in the schedule

Conflicts

A conflict is a constraint violation

Temporal constraint violations

- *A* starts at end of *B* *A* starts at 100, but *B* ends at 50

Resource usage violations

- Power maximum is 50 but there are 2 activities that use 30 each at the same time

State usage violations

- *B* requires that the filter be blue but the filter is red during *B*

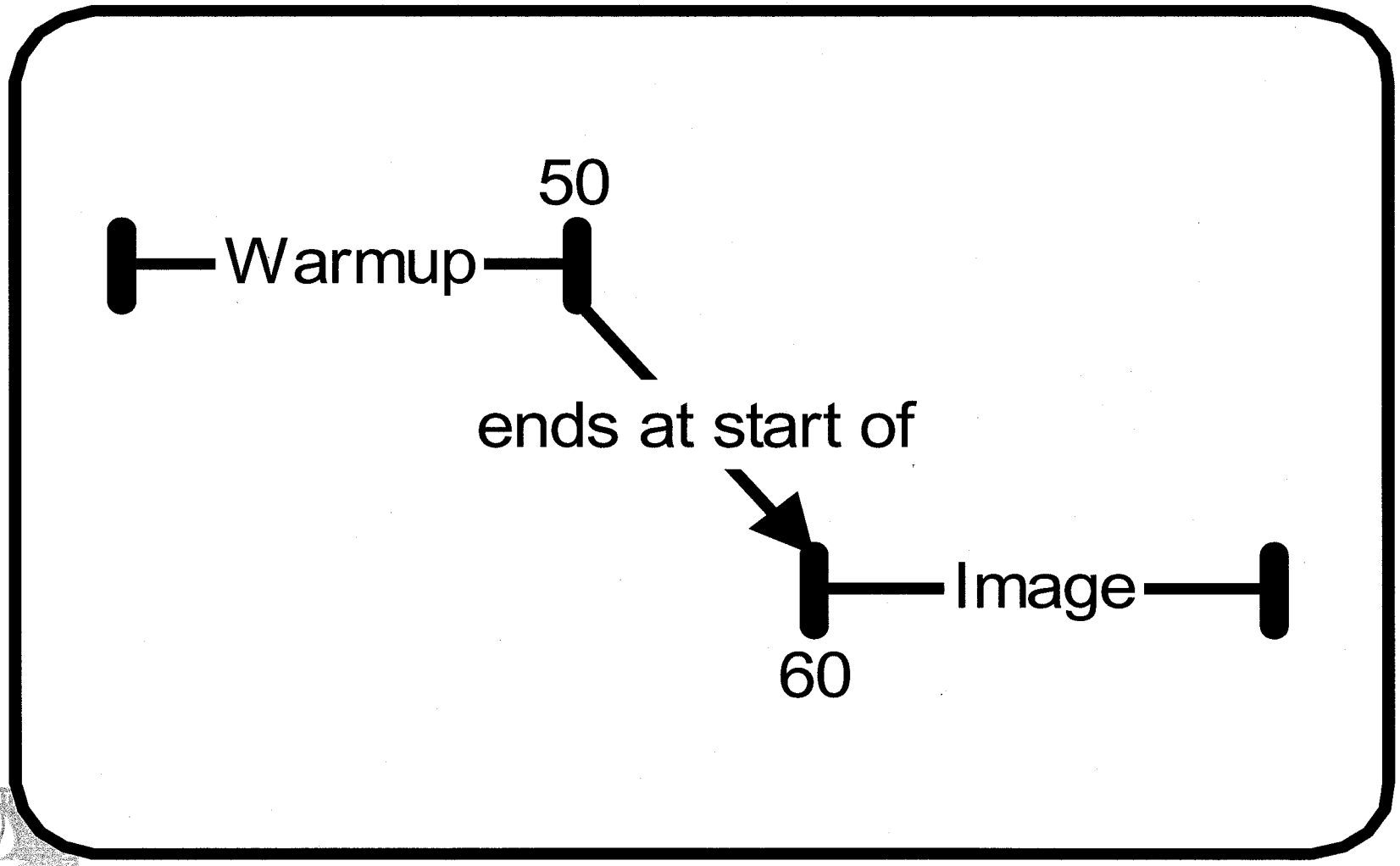
State transition violations

- The light can go from red to green, green to yellow, and yellow to red but *C* causes the light to go from green to red

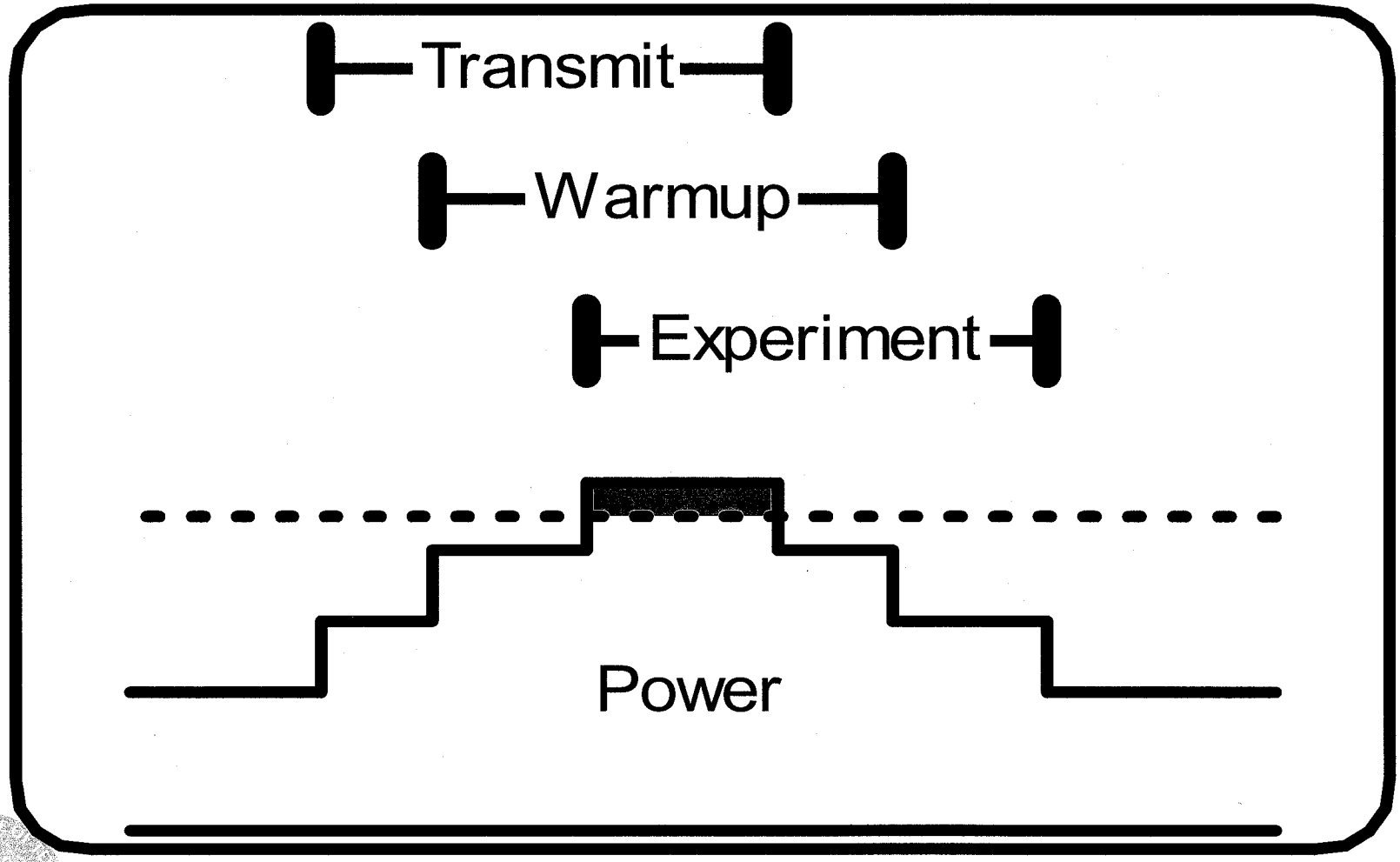
Dependency violations

- x = 1, y = 2 z = (x + y) but currently z = 0

Temporal Constraint Violation



Resource Violation Example



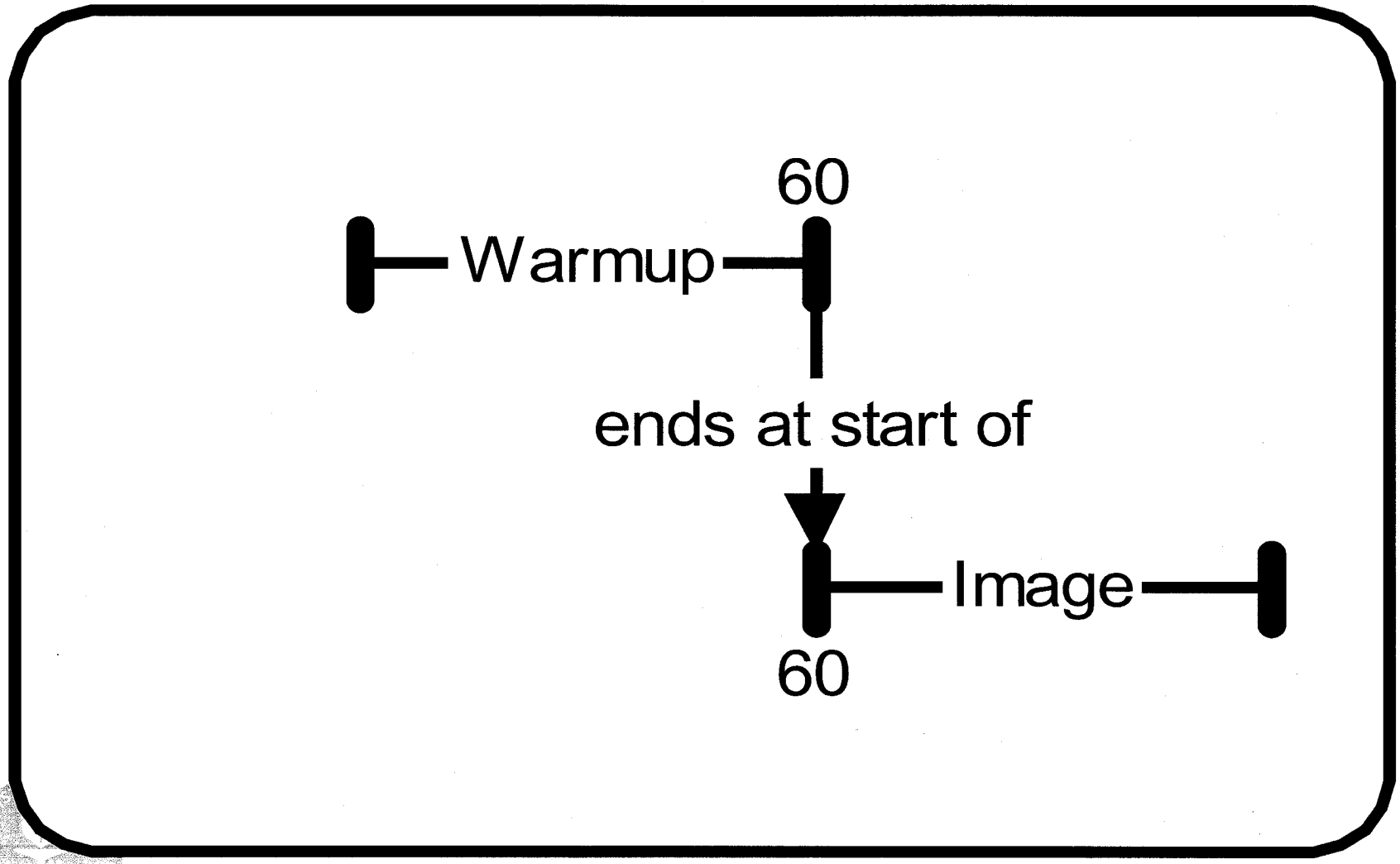
Repair Methods

Repair Method: an operation that leads to fixing the conflict

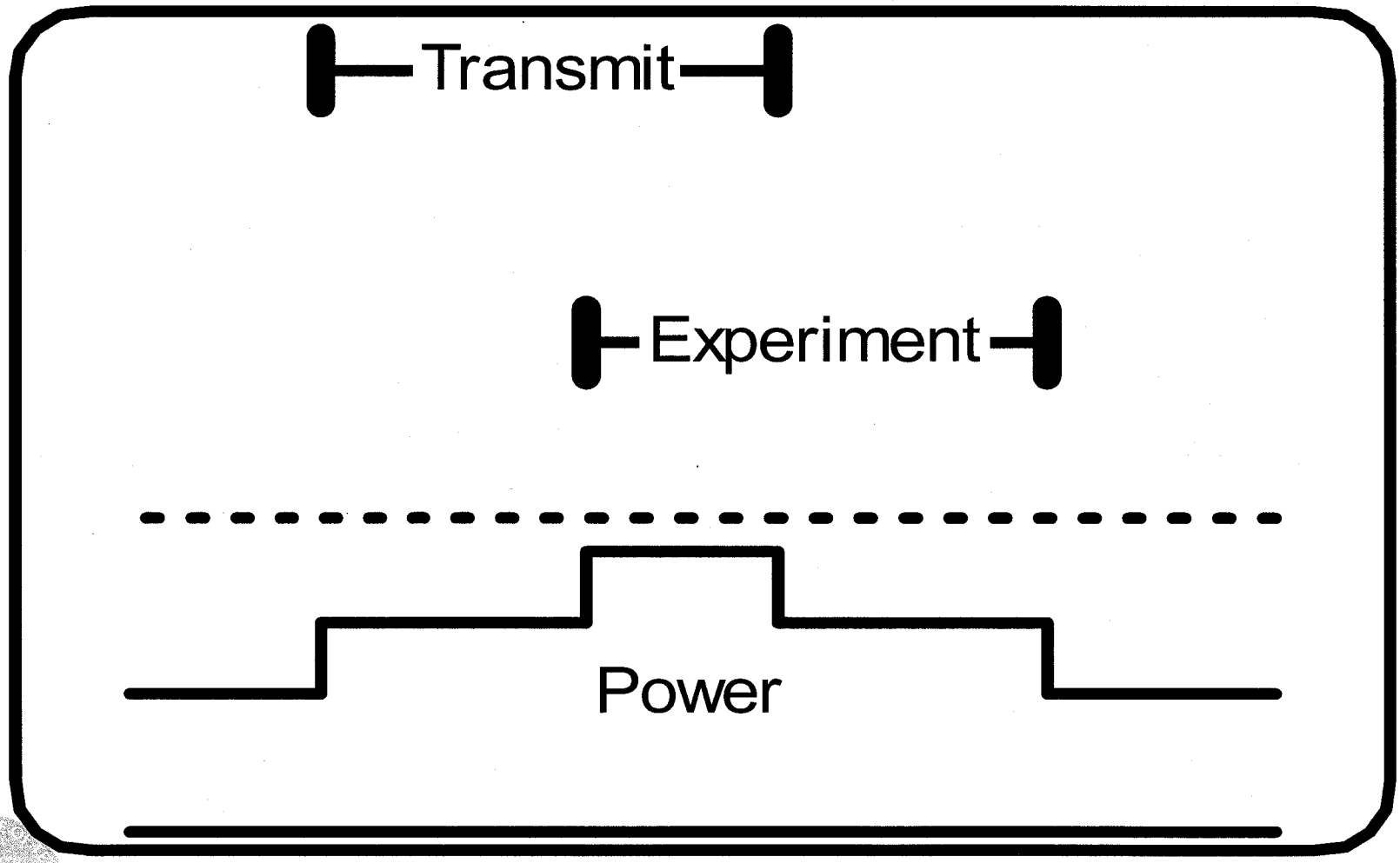
Depends on the conflict and the model

- Some repair methods are useless for some conflicts
- Some repair methods are not possible with some models

Move Repair Method Example



Delete Repair Method Example



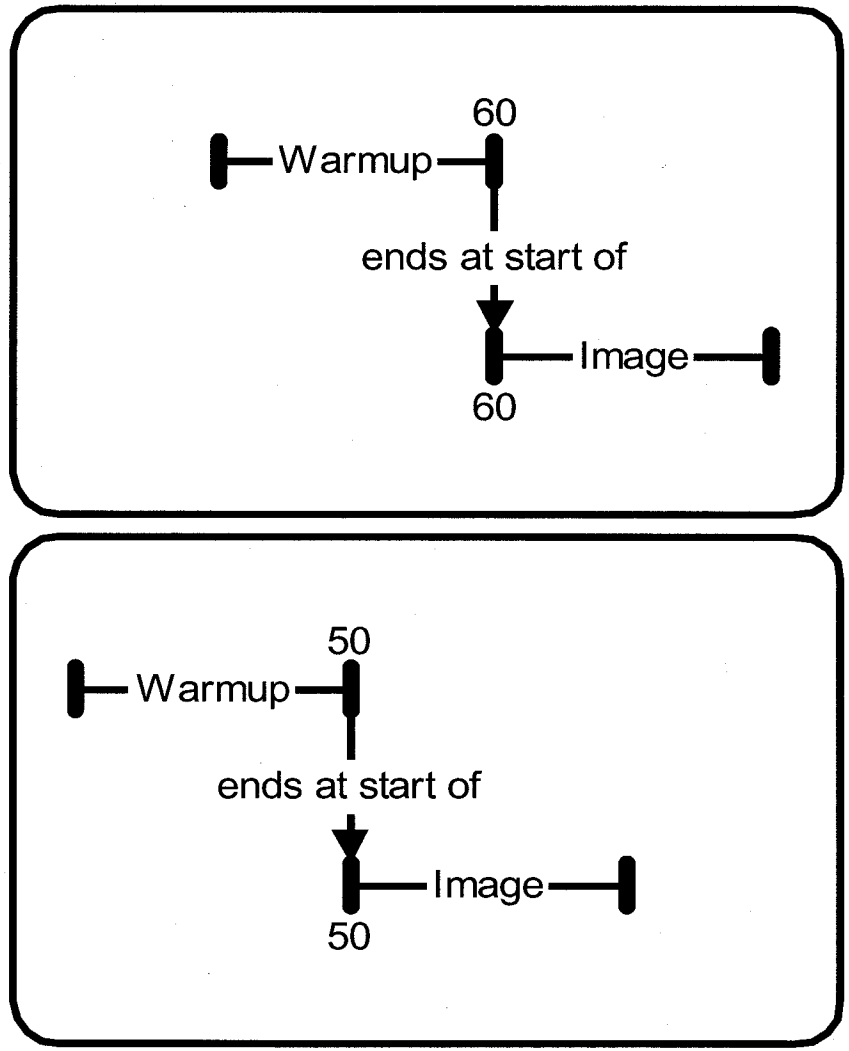
Choice Points

Choice points are the forks in the road to finding a solution.

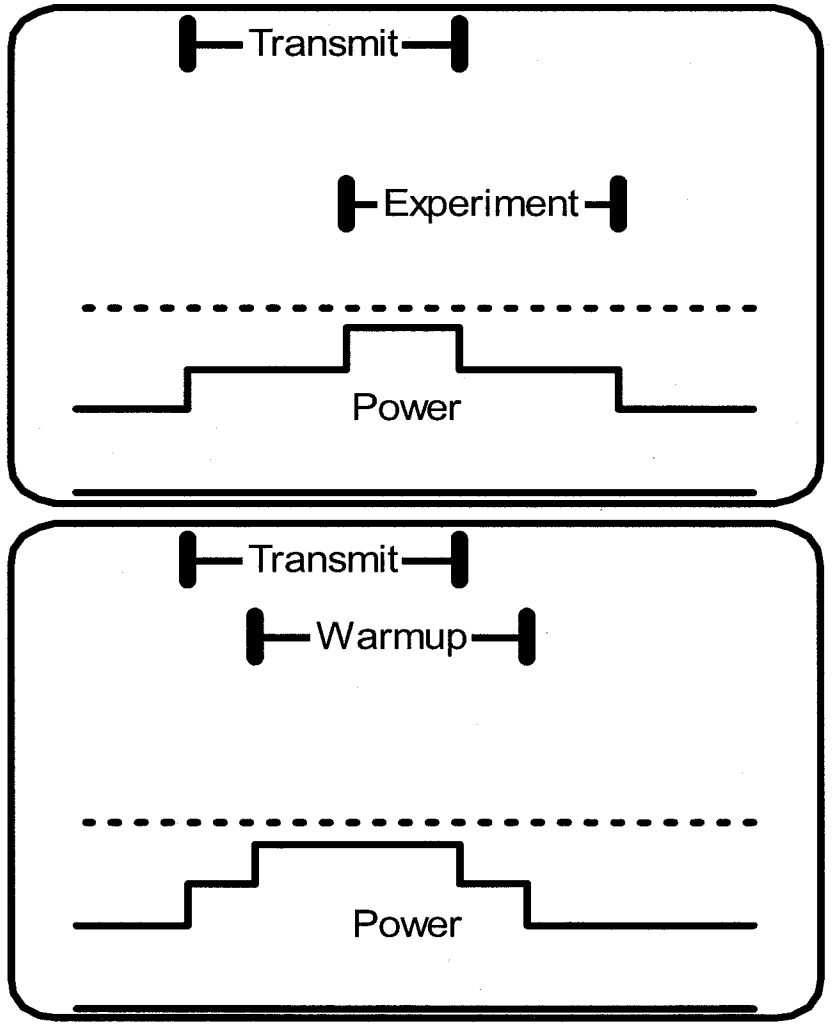
At each point, we must choose among the options available.

Instead of keeping track of all of our choices, we opt to make choices quickly and keep moving forward, possibly causing other conflicts.

Move Choice Point Example



Delete Choice Point Example



Heuristics

Heuristics guide the search for solutions by expressing a preference for one choice over others at each choice point

Move

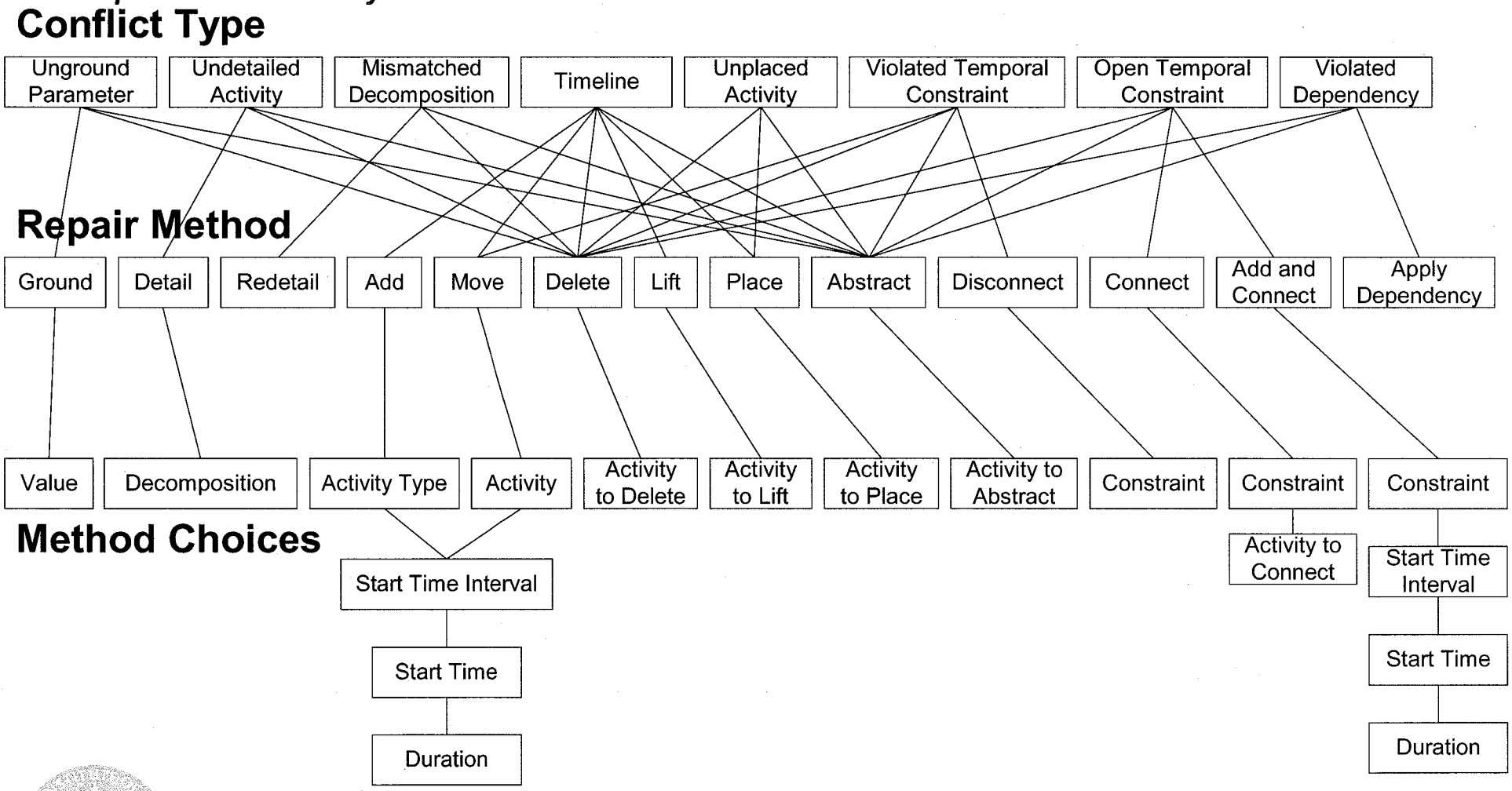
- Earliest activity, latest activity

Delete

- Earliest activity, latest activity, least valued activity

The Repair Search Tree

Jet Propulsion Laboratory



Packing

Allows for fitting more activities into the schedule or for making elbow room for existing activities.

Earliest

- Pack activities as early as possible without causing a conflict

Latest

- Pack activities as late as possible without causing a conflict

Loosest

- Attempt to space out the activities evenly over time

Constraint Propagation Planning

Jet Propulsion Laboratory

- Constraints, timelines, tokens
- Constraint propagation
- Search

Ari K. Jónsson
NASA Ames Research Center

Advanced Topics

- Generalized Timelines
- Operations Research Approaches

Generalized Timelines

Motivation

Spacecraft operations require modeling complex modeling beyond what traditional planning and scheduling systems, examples include:

- Pointing quaternion (point the HGA at earth; and point the camera is at Jupiter; and avoid any thermal or boresight avoidance angles)
- File Systems
- Power with solar panel charging, battery, plus current draw.

These are not readily expressible as monotone values with domains over which constraints can be levied (i.e., more "constraints" can be less constraining than fewer), yet verifying any solution is still straightforward once a solution is arrived at.

Required Operators for Generalized Timelines

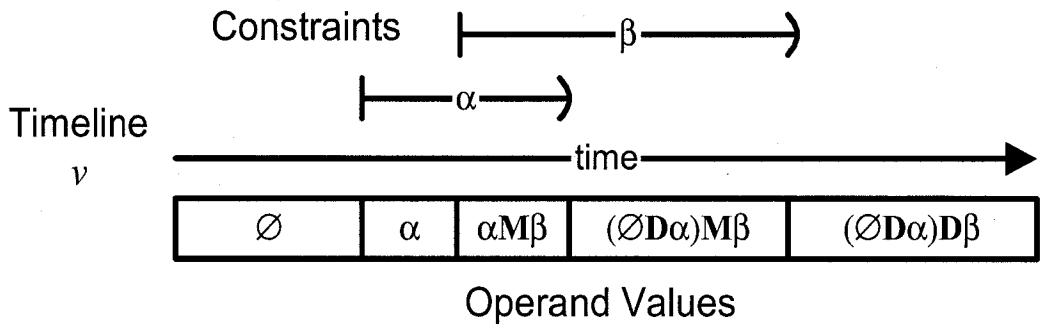
A merge operator ($x\mathbf{M}y$, $x \in O$, $y \in O$) that is associative and commutative. We use this to merge operand values together.

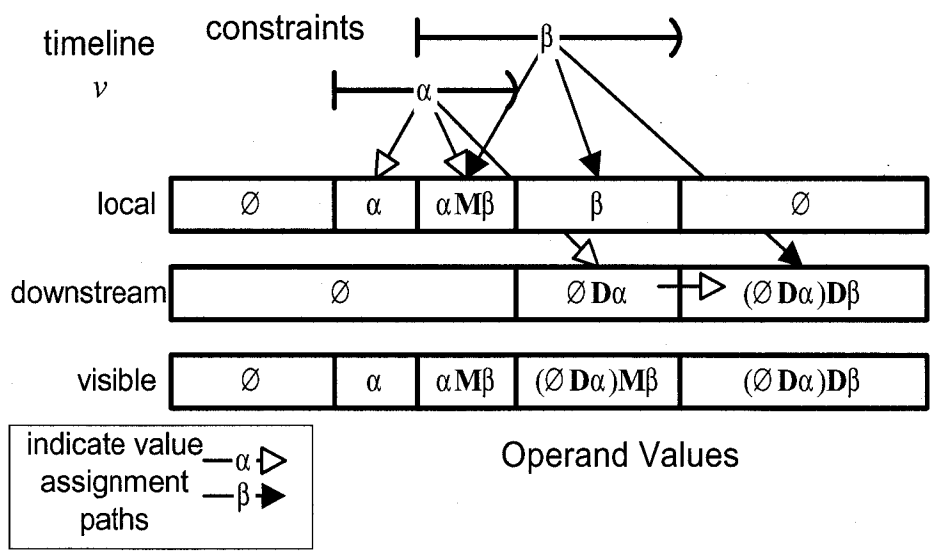
A value propagation operator ($x\mathbf{D}y$, $x \in O$, $y \in O$) used for calculating the "down-stream" effects of a value x on the value y given the context of the current value y. This is used to compute values after delta constraints.

A consistency checking predicate **OK**(x) that is used to validate that a particular operand value is not violating any constraints associated with it.

A consistency checking predicate **XOK**(x,y) that is used to validate that a transition from the first operand value (x) to the second (y) is not a violation of any constraints.

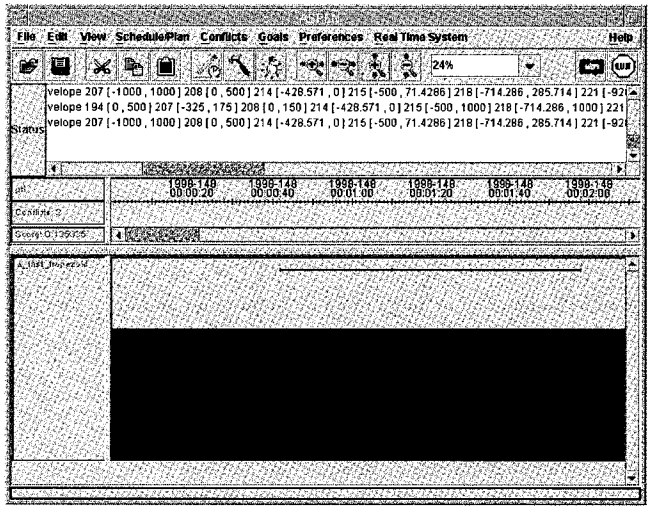
Abstract Timeline



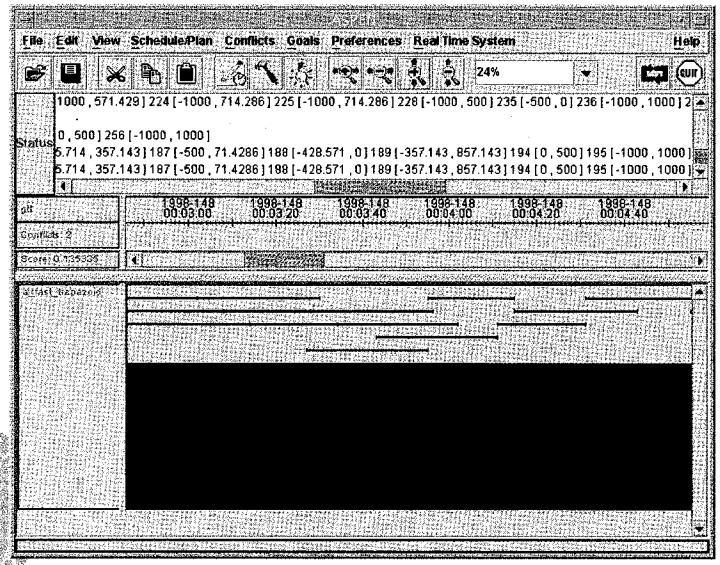


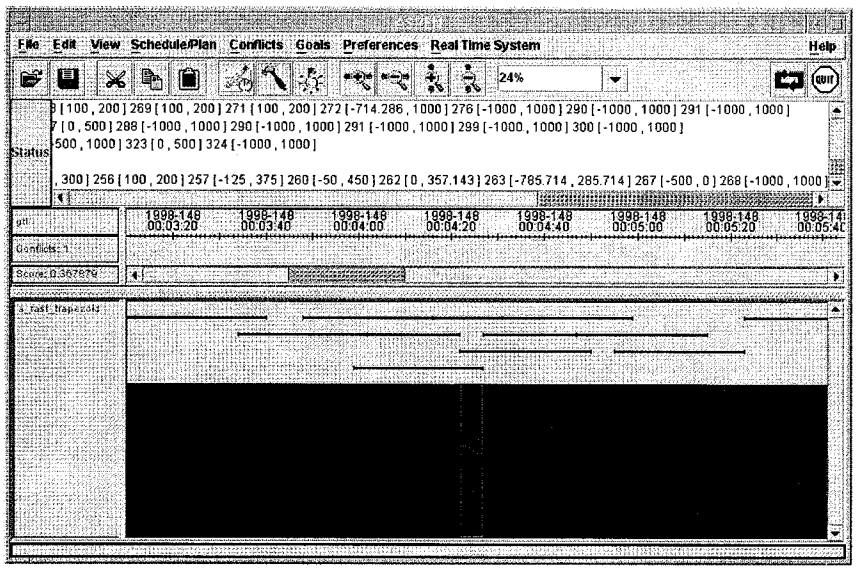
Temperature Envelope Example

Jet Propulsion Laboratory



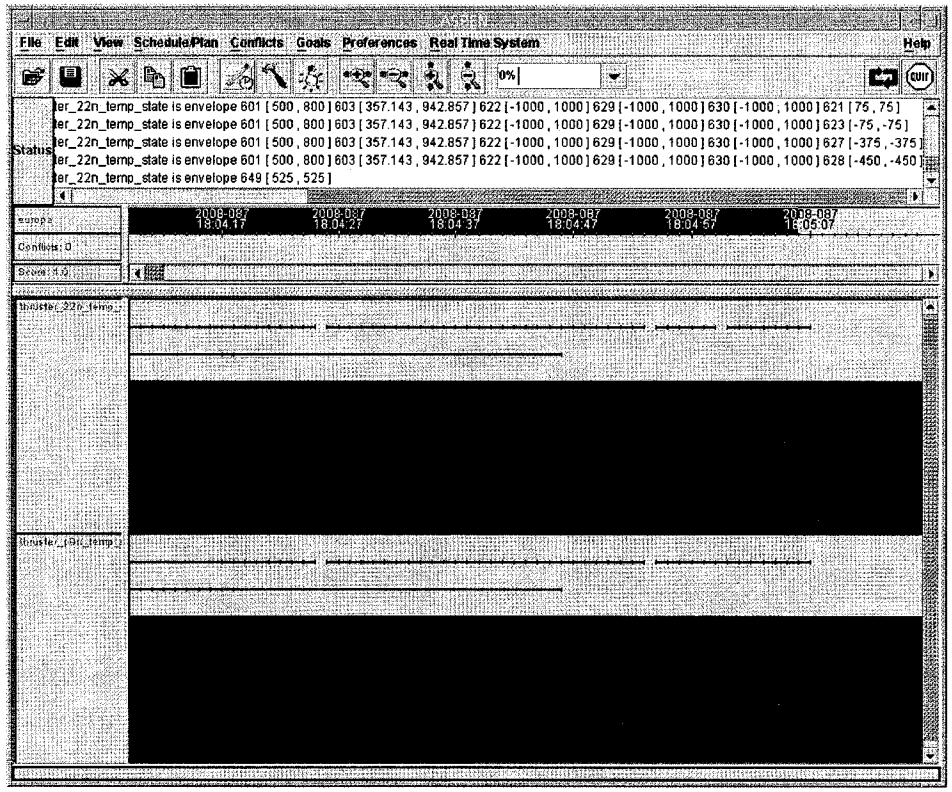
Examples of a single temperature constraint over time, a collection of temperature constraints, and a similar collection of constraints with a constraint violation.





Temperature Envelope Example

Jet Propulsion Laboratory



An execution of the schedule for the Europa Orbiter. Visible are the thruster temperature timelines, each with measurements visible (the blue diamonds.) The black bar above the timelines represents the current time.

OR-based Approaches

- Operations Research (OR) offers a number of techniques for solving specialized classes of problems (e.g., linear programming, integer linear programming, classes of optimization problems).
- Often OR solutions involve coercing a problem into one of these frameworks
- OR techniques can often be used to solve subproblems or larger scheduling problems

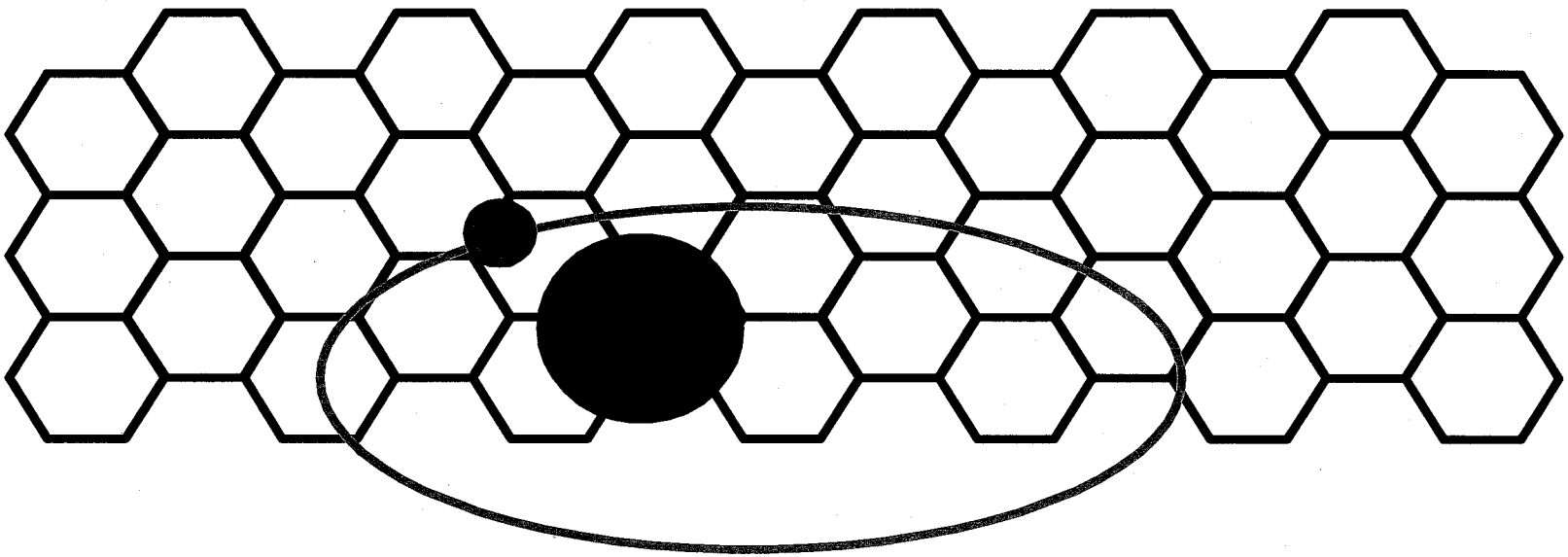
OR-based Examples

Space Interferometry Mission (Negative Example)

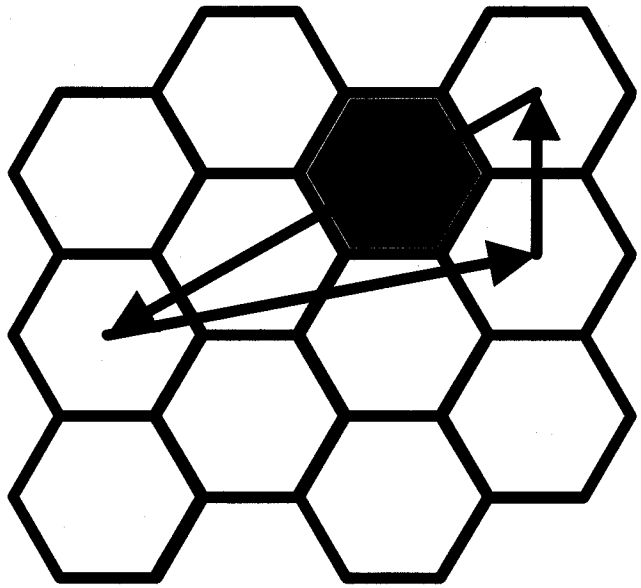
Nadir Observation Scheduling (Positive Example)

SIM (Space Interferometry Mission)

- Acquire image pairs of the celestial sphere
- Avoid “bright-spots”
- Keep in mind limited resources such as downlink capacity, memory, power, energy, instrument set-up, etc.

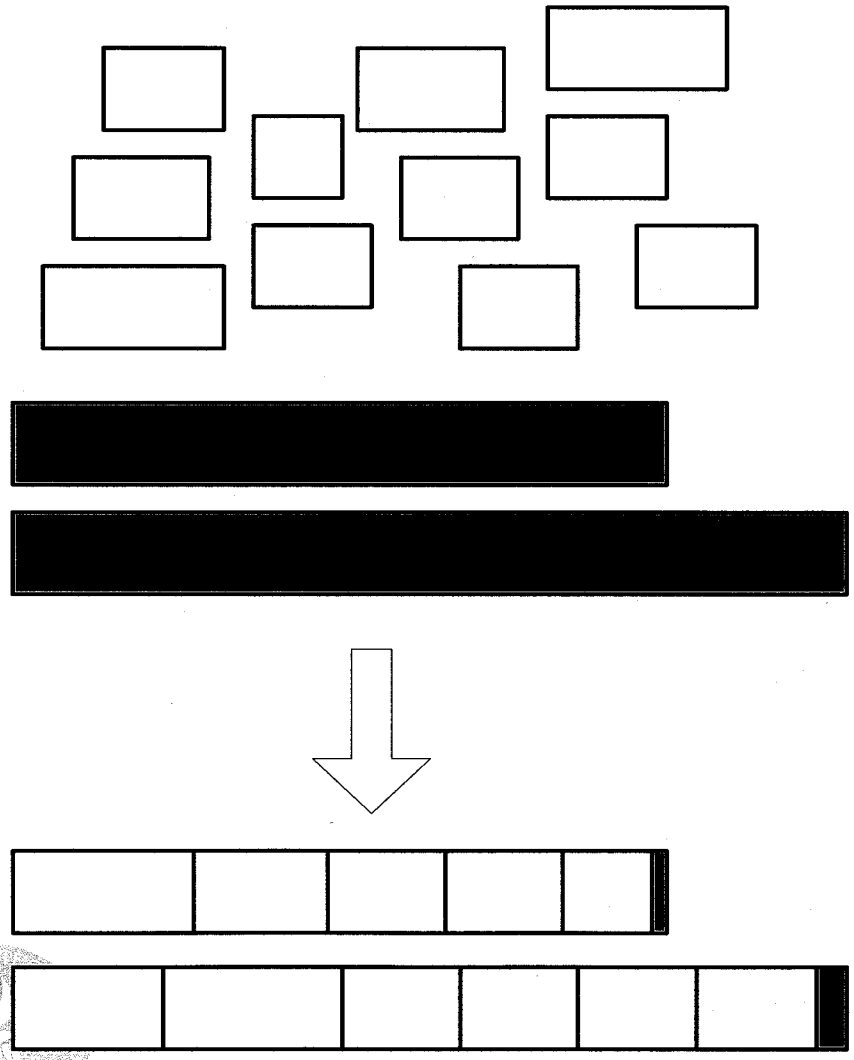


Traversing from Tile to Tile



Therefore, the SIM problem involves the *Traveling Salesman Problem*

Fitting Observations to Downlinks



Therefore, SIM has elements of bin packing (or k-knapsack, depending on objective function)

Problem Description

Different Common Observation Scheduling Objectives

- Make as many observations as possible given constraints
- Gather as much data as possible given constraints
- Make at least n observations as soon as possible
- Gather at least m data as soon as possible

Involves the Traveling Salesman Problem with Time Windows and the Bin Packing Problem

Solution Approach

Approach – use existing solvers in combination

Two types of Solvers: IP and IR

Integer Programming

- Express both problems as integer programs
- Combine the integer programs
- Combine the search strategies
- GLPK

Iterative Repair/Optimization

- Express both problems as “iterative repair” problems
- Combine the problems
- Combine the search strategies
- ASPEN

Mixed Integer Programming

Linear Programming:

Continuous real-valued Variables

Constraints in the form of linear inequalities over the variables

Objective Function in the form of a linear equation over the variables

Integer Programming– all of the variables must be integer

Mixed Integer Programming– some of the variables are integer, others are linear

Comparative Performance

Problem Generation

- 100 problems
- 50 observations, 25 windows each
- 16 downlinks
- Asymmetric transition times from tile to tile
- Highly variable observation memory requirements
- Low variance of downlink capacity
- Slightly less than 4K variables and 12K constraints in the IP formulation

Hardest for Iterative Repair/Optimization

Depth First Branch and Bound Search

Depth first search.

Don't take any branches that are not as good as your current best answer.

Estimate branch cost using your current cost plus a heuristic estimate of taking the branch.

This provides provably optimal answers if the heuristic estimate never over-estimates cost (is admissible).

Cost can be inverted to mean reward.

Branch and Cut Search

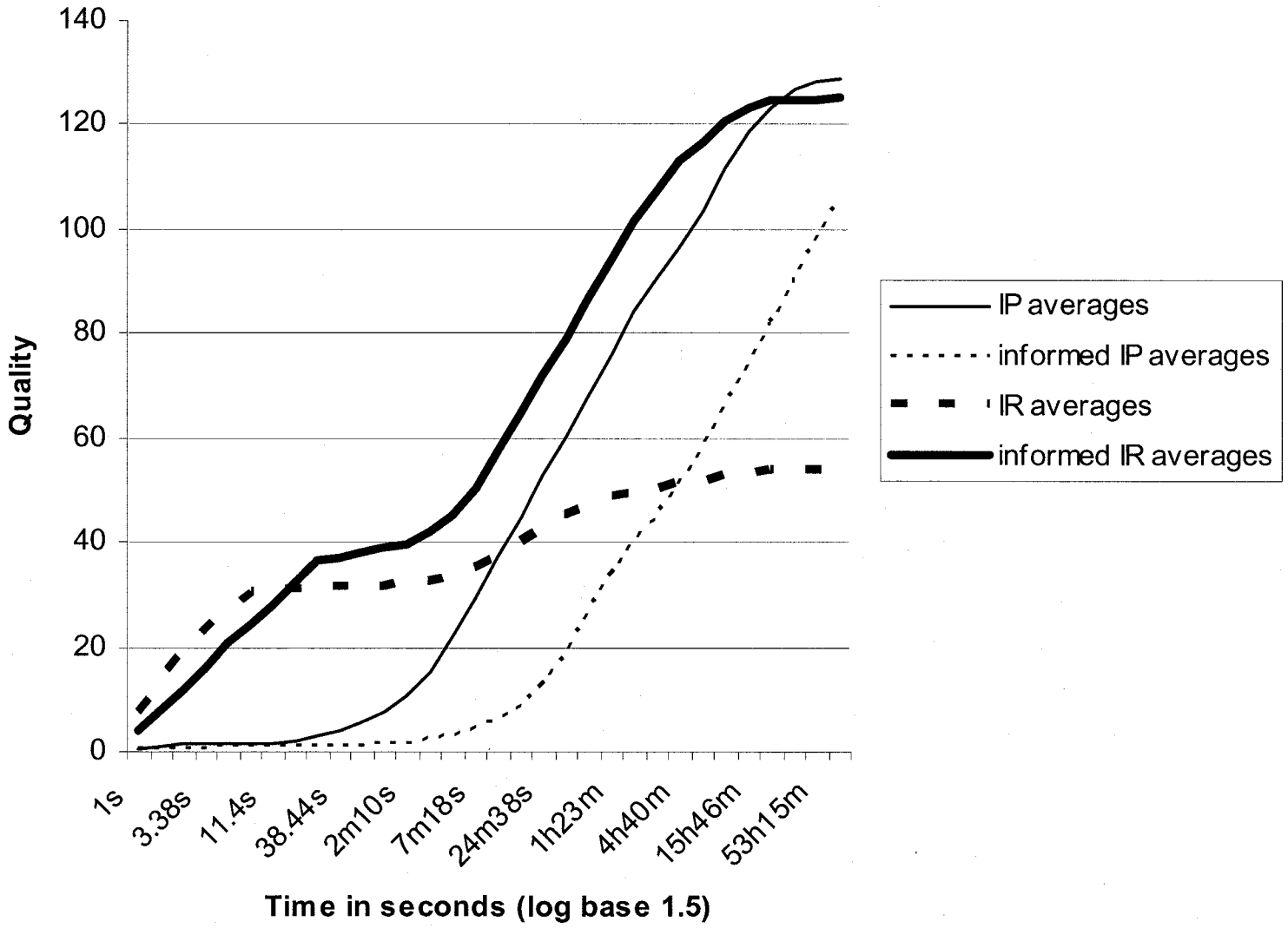
Similar to branch and bound, but:

At each point, we may decide to improve our heuristic (the linear programming relaxation) by generating new rows (or columns, in the dual formulation) that improve the LP heuristic performance.

If the new row (inequality) removes part of the polygon represented by the current set of inequalities, it has produced a cut (it is facet-inducing).

Knowing that an inequality produces a cut is NP-complete in the worst case.

Comparative Performance



Empirical Performance Results

Integer Programming (alone)

- Not surprisingly poor short-term performance, Optimal long-term performance

Informed Integer Programming

- Quite surprisingly poor performance when compared with IP alone

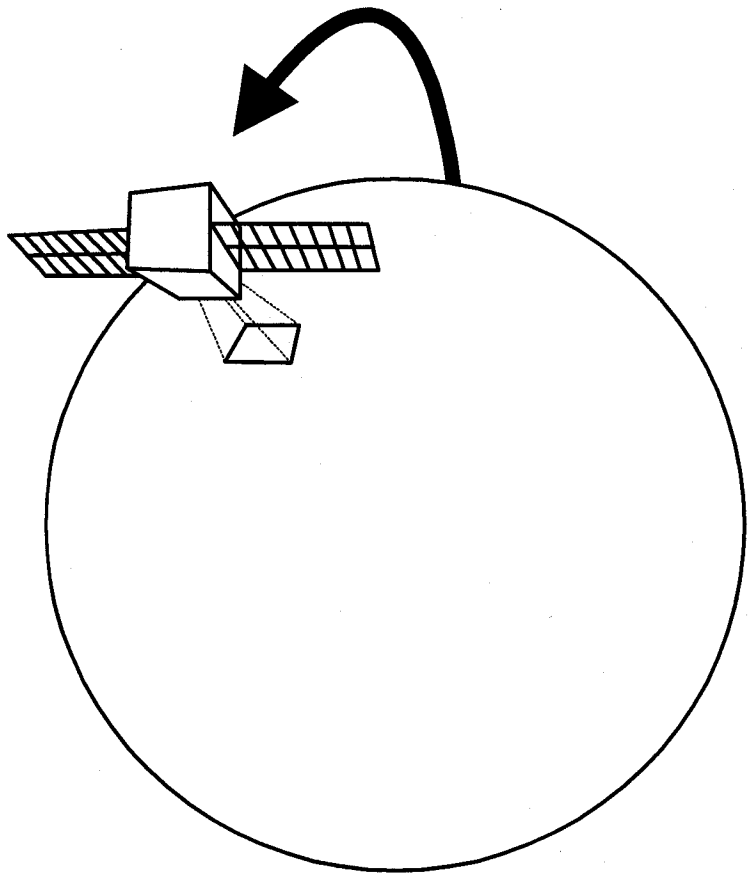
Iterative Repair (alone)

- Good short term performance, Not surprisingly poor long term performance

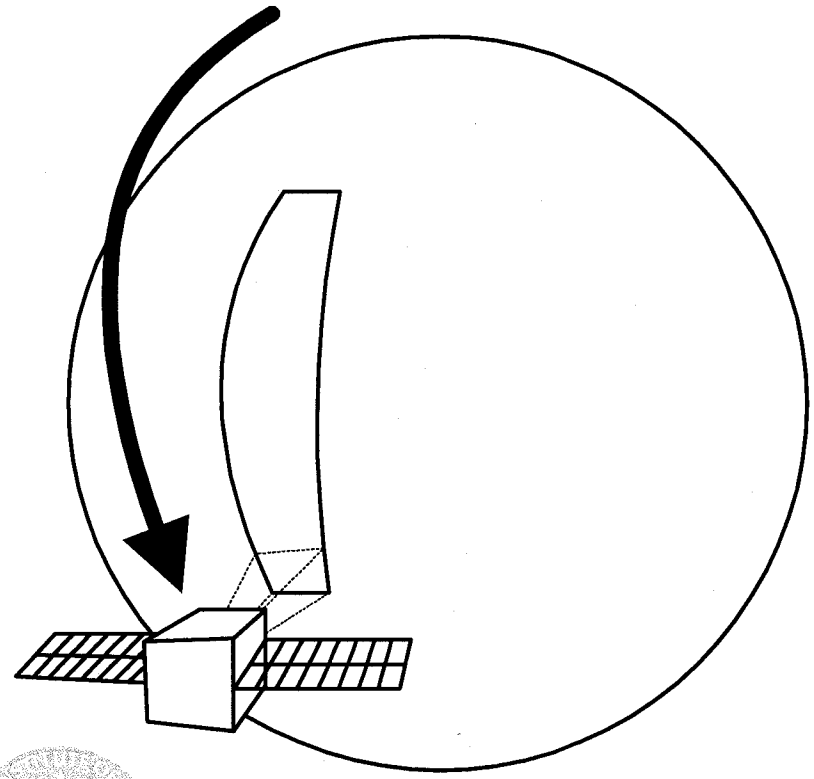
Informed Iterative Repair

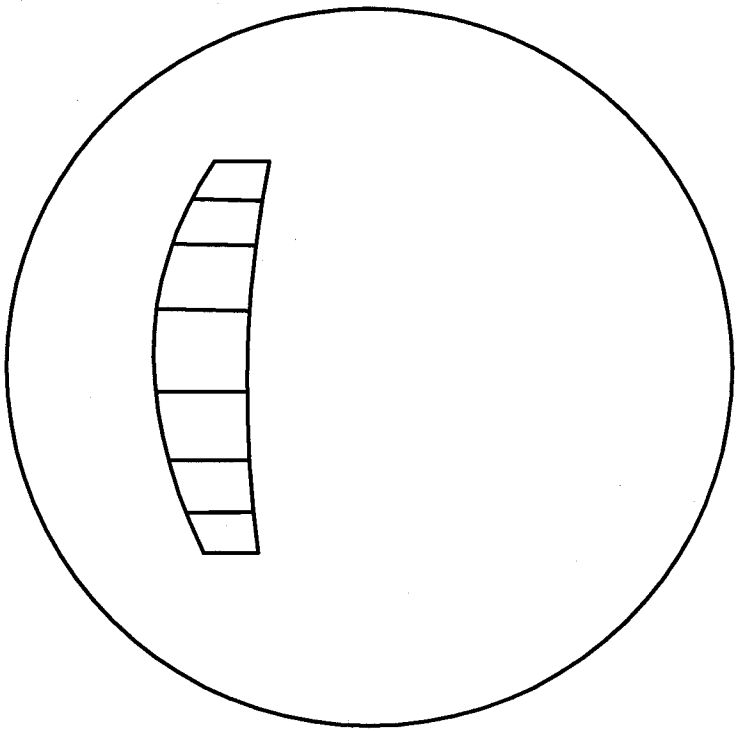
- Fairly good performance overall

Nadir observation scheduling

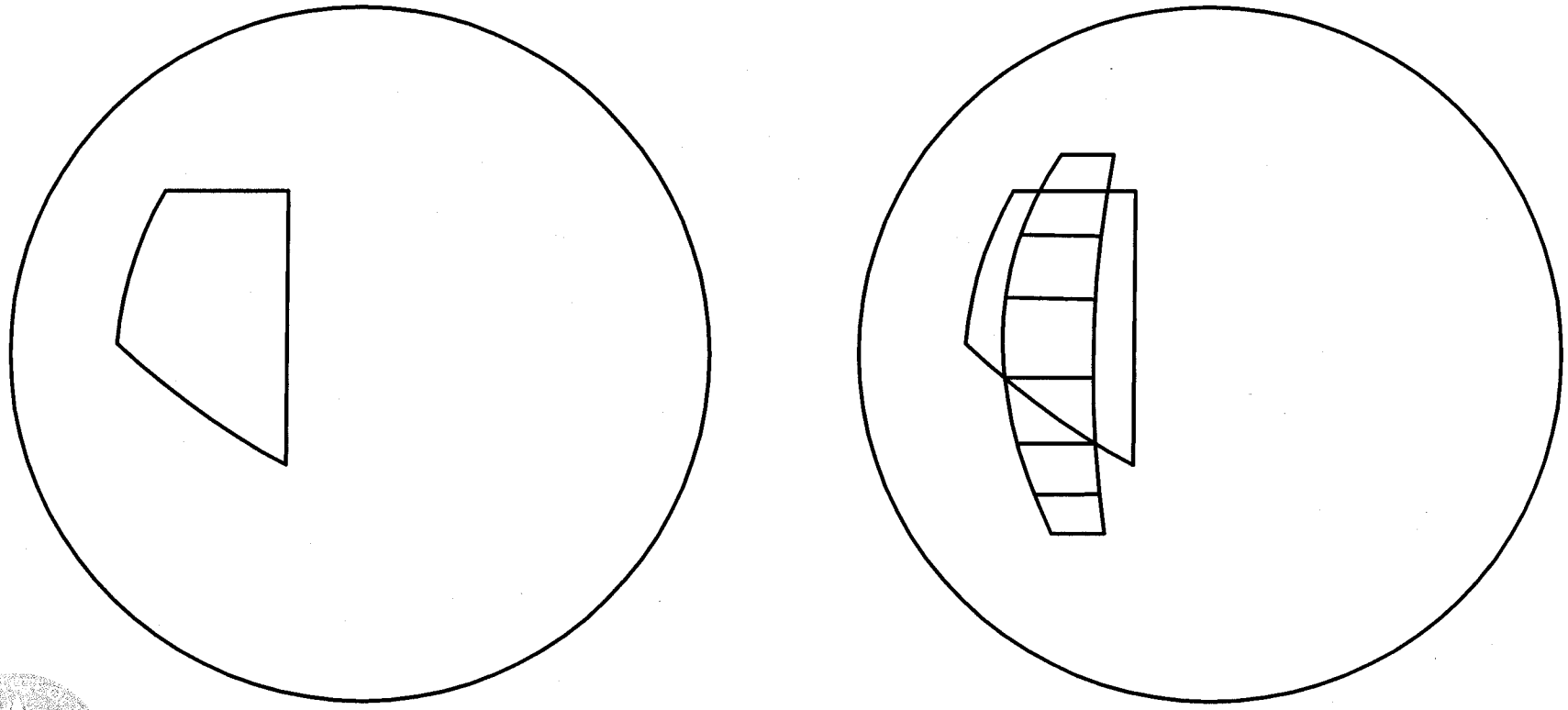


Swath and segments

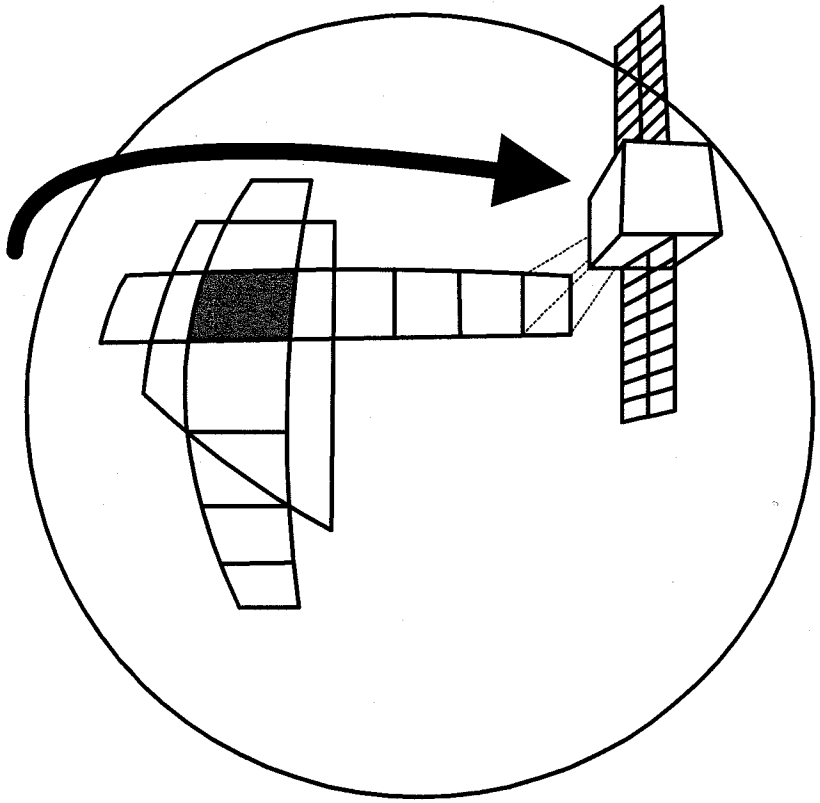




Target and segments



Overlapping Segment



Memory and communications

Onboard memory

- Fixed
- Consumed on imaging, released on downlink

Communications – downlinks

- Communications occur at all downlinks
- Downlinks occur over intervals and can communicate a limited capacity

Carrying data over a downlink

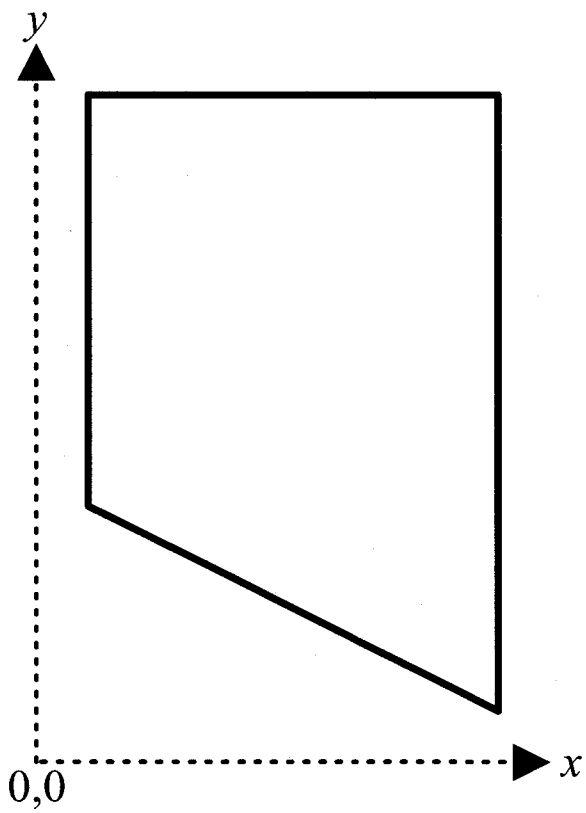
- Allowed
- Not allowed

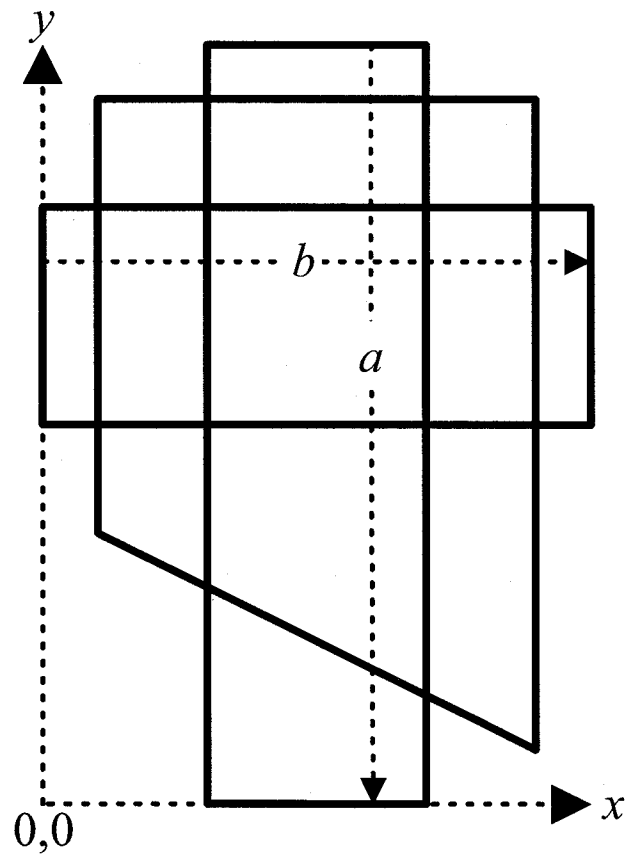
Swath segment selection problem

SSSP: chose segments that result in the gathering and subsequent downlinking of target data without overrunning our available memory or communications capacities

SSSP example

Swaths a and b

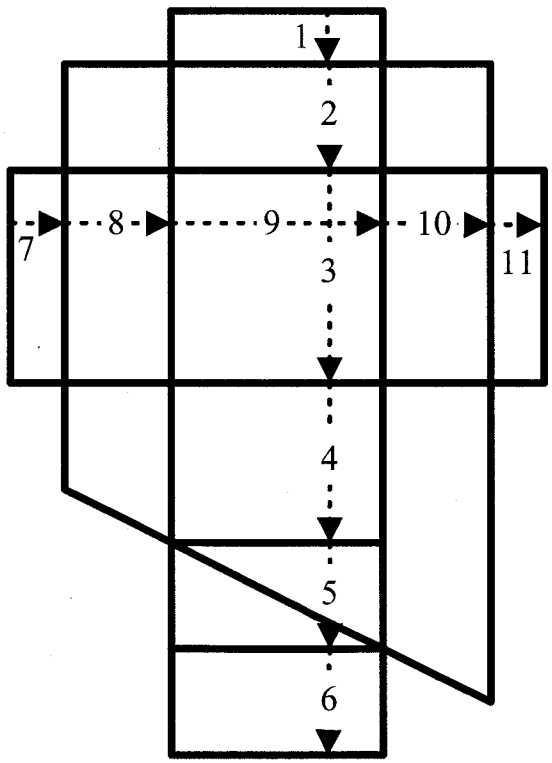


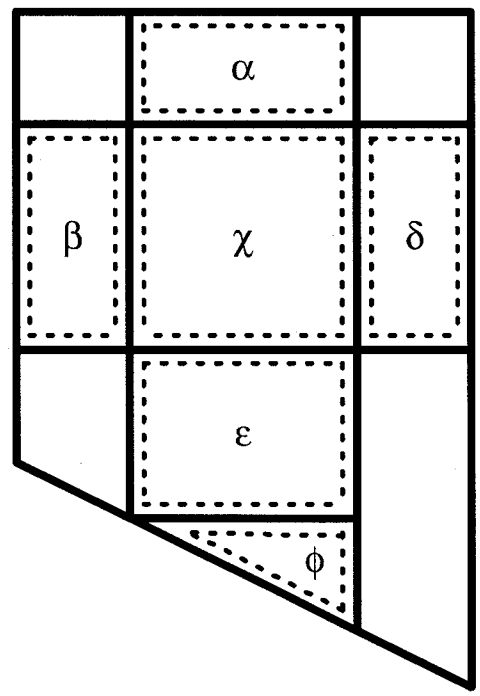


SSSP example, continued

Segments 1 – 11

Shards $\alpha - \phi$, shards ARE targets





SSSP example

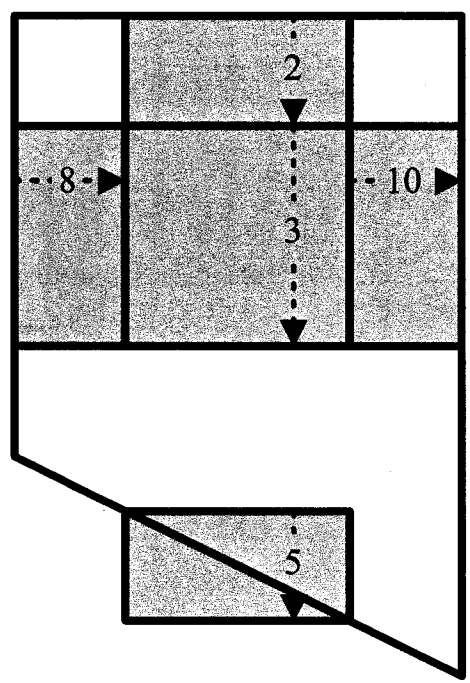
segment	area
1	4
2	8
3	16
4	12
5	8
6	8
7	4
8	8
9	16
10	8
11	4

shard	area
α	8
β	8
χ	16
δ	8
ε	12
ϕ	4

SSSP example, suboptimal solution

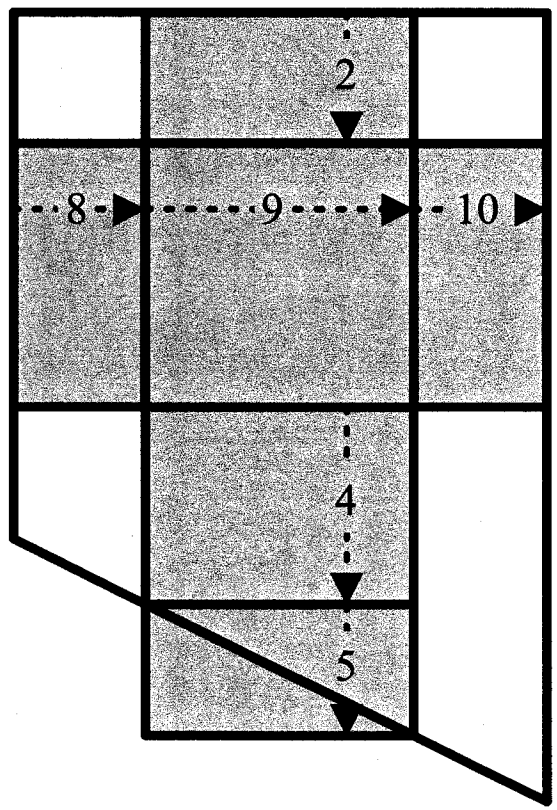
Downlinks are after each swath for 32 units each, memory capacity is 32

Solution: {2, 3, 5, 8, 10} = 44 units



SSSP example, optimal solution

Solution: {2, 4, 5, 8, 9, 10} = 56 units



What is the SSSP?

Q: Is there an already well-studied combinatorial optimization problem that solves this?

IP, of course, given a reduction = straw-man

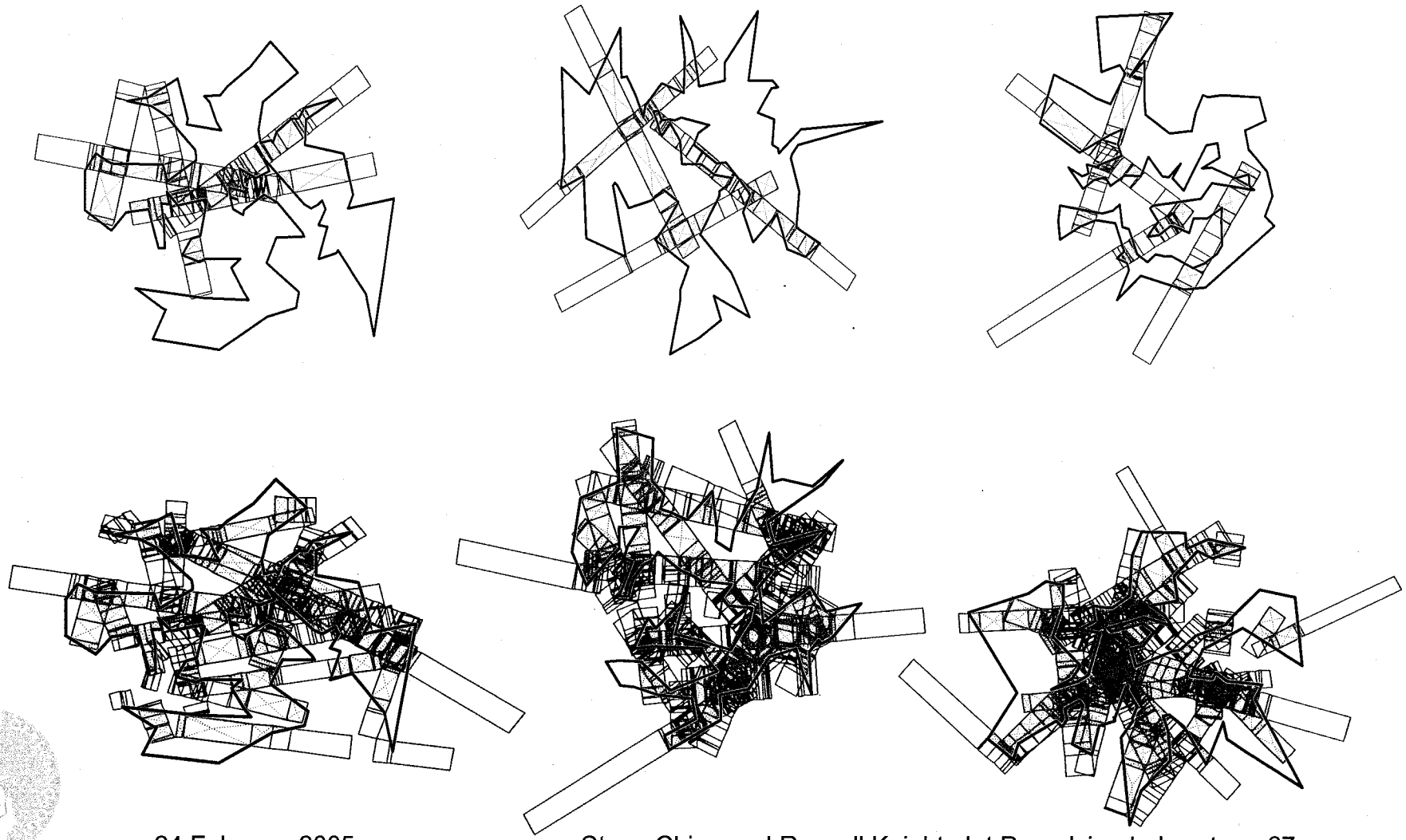
Minimize the number of segments that collect all shards = set cover

Maximize the downlinked segment area (ignoring shards, memory, and time) = k-knapsack

Collect all segments (ignoring shards) = Pickup and delivery with capacity (and a fixed route)

A: No, not really

5 and 20 swaths



Solution techniques

Greedy

- The ASTER algorithm

Depth-first branch and bound (DFBnB)

- Mixed Integer Program (Always try this first!)
- Flow network heuristic and its relationship to the linear relaxation for IP.

Greedy

For each swath, greedily include the "best" segments, until memory is exhausted.

Best is determined by the highest ratio of reward/capacity cost.

Ties are broken in order of time (earlier is better).

VERY fast!

Can be sub-optimal

- E.g.,Yields the sub-optimal solution in our example.

Depth-first branch and bound

DFBnB

- Node ordering heuristic: which subset should I decide whether or not to include?
- Reward estimating heuristic: given my current selected subsets, how good can my result be?

IP formulation

0-1 Integer variables are subset selection variables.

Node ordering heuristic: the variable that is closest to either 1 or 0 after computing the linear programming relaxation.

Reward estimating heuristic: the linear programming relaxation.

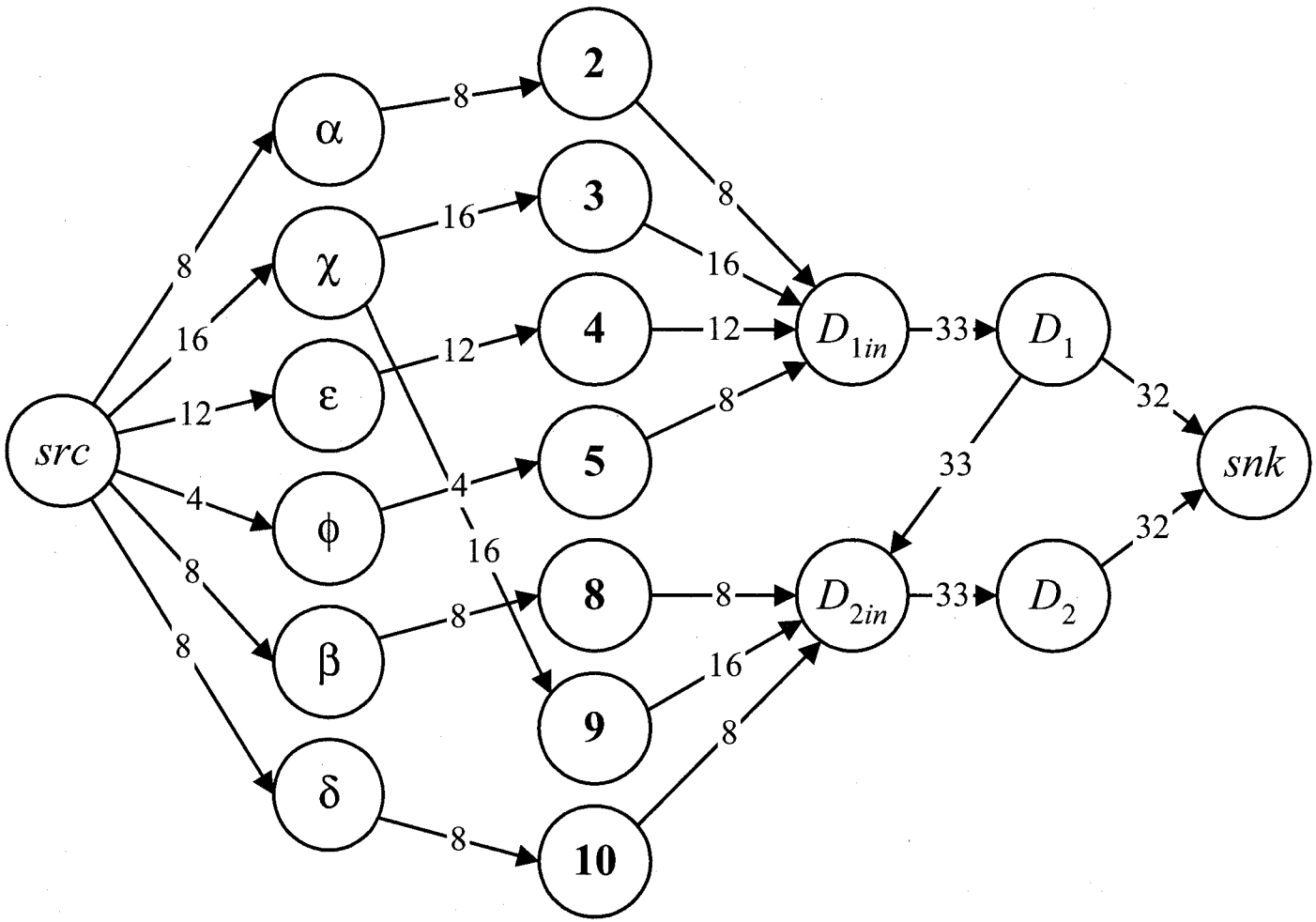
Network flow formulation

Select (or deselect) subsets.

Node ordering heuristic: choose the subset according to the best reward/cost ratio, breaking ties at random.

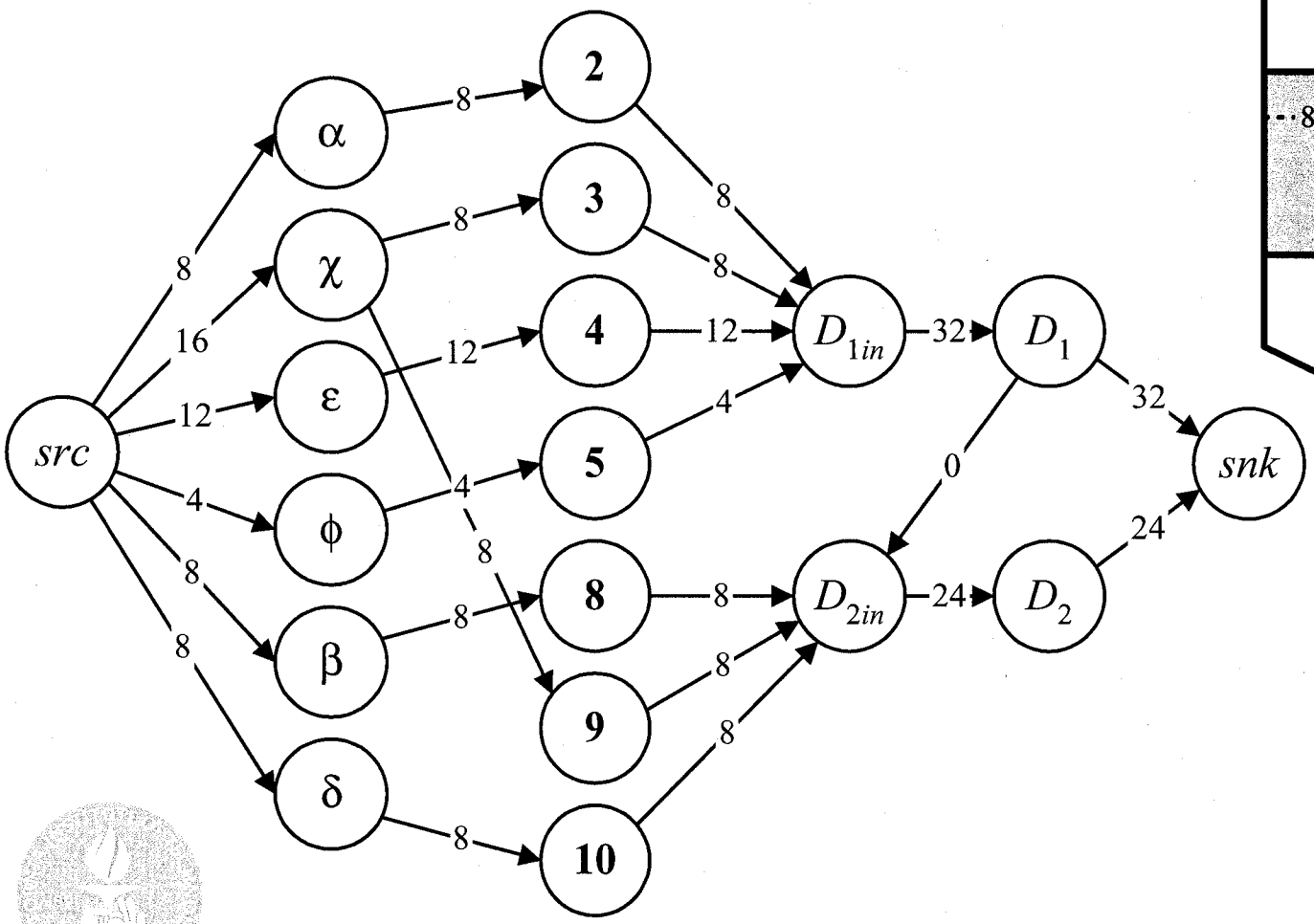
Reward estimating heuristic: the network flow relaxation.

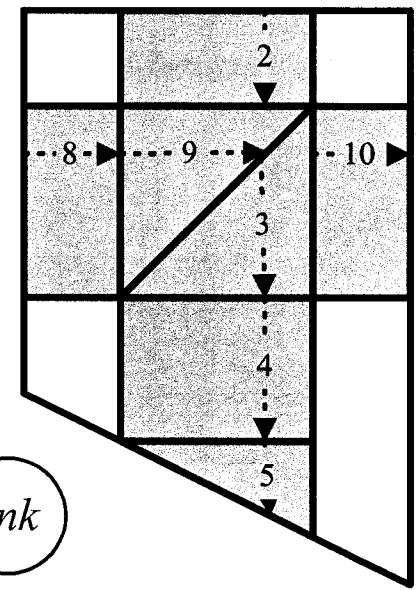
Network flow formulation



Network flow relaxation solution

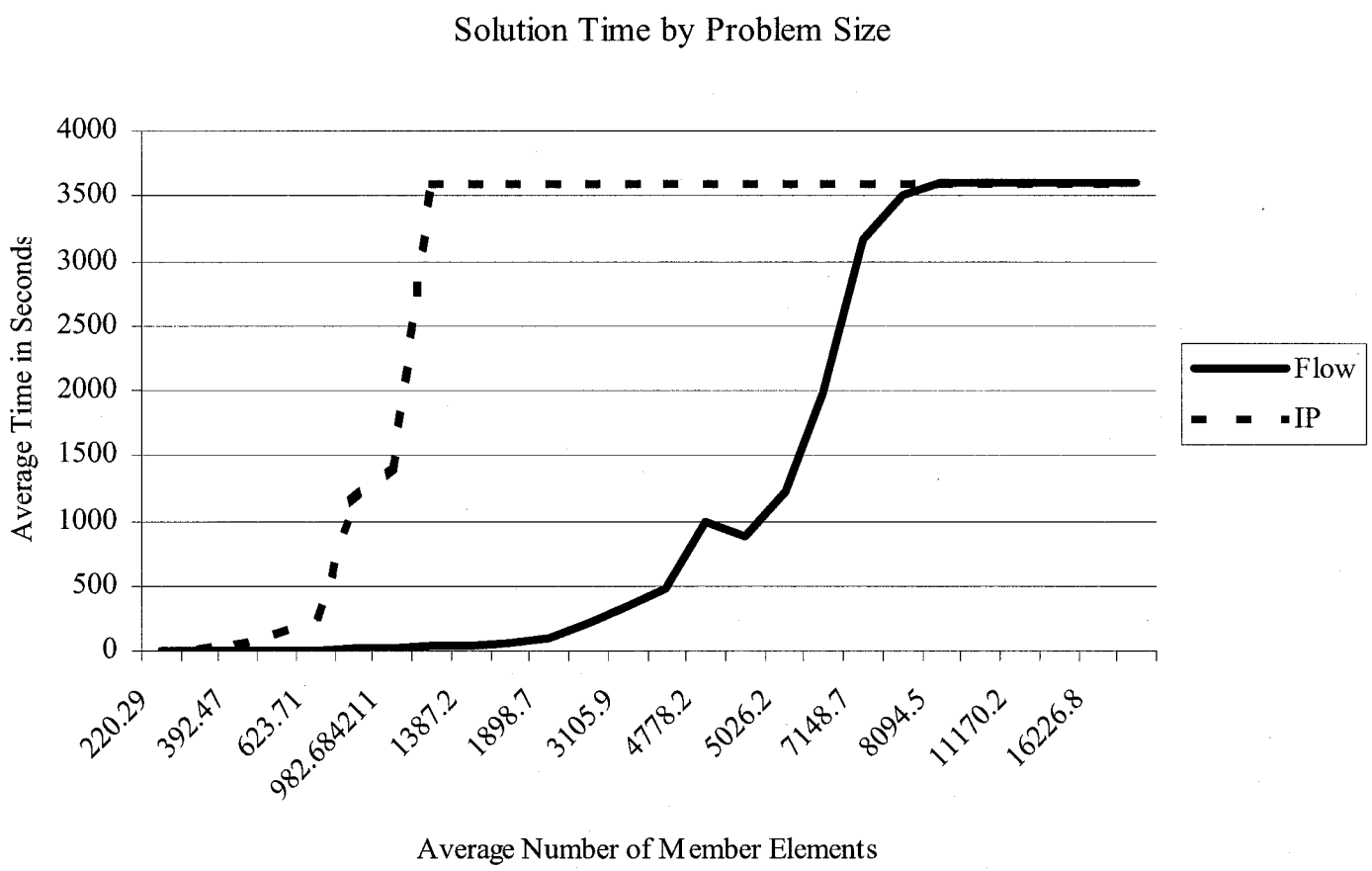
Jet Propulsion Laboratory



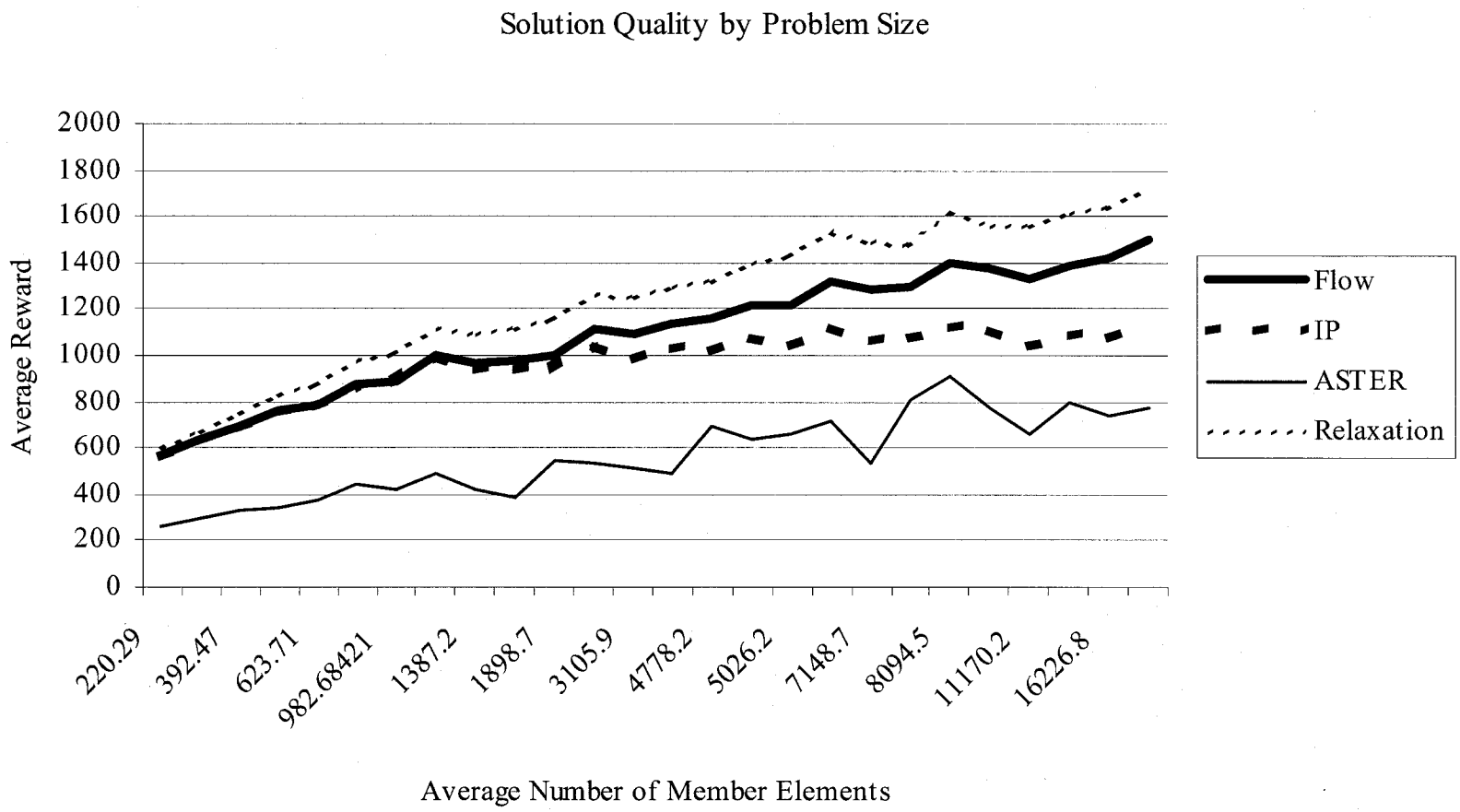


Comparative Speed

Note that ASTER required less than 1 second for all instances



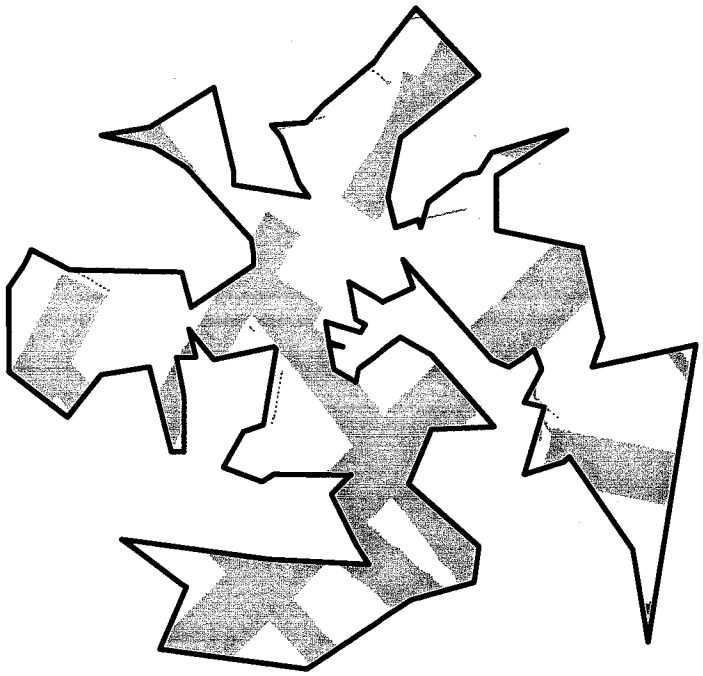
Comparative Quality

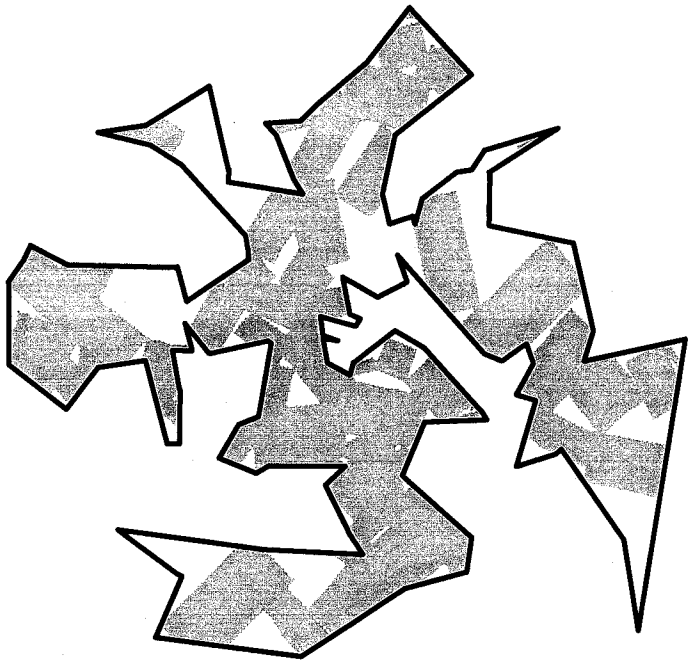


Comparative Coverage

ASTER solution area = 819.585

Flow solution area = 1383.29





Reinforcement Learning for Method Applicability

[Zhang & Dietterich JAIR] apply TD(λ) reinforcement learning to learn applicability conditions for schedule repair operators

Used Space Shuttle Ground Processing schedule data

2 Concepts learned are REASSIGN-POOL (pick new resource assignment) and MOVE (move activity)

method uses a sliding window over schedule

- individual features on activities and resources are utilization, numbers of temporal constraints, etc. # of windows in conflict, etc.
- output of network is an action (MOVE, REASSIGN-POOL) to be applied to the schedule

Research Trends and Directions

- Finite-capacity scheduling under more complex constraints and increased problem dimensionality
  - subcontracting, overtime, lot splitting, inventory, etc.
- Integrated planning and scheduling
- Mixed-initiative frameworks
- Management of uncertainty (proactive and reactive)
- *Autonomous agent* architectures and distributed production management
- Integration of machine learning capabilities
- Wider scope of applications
  - » analysis of supplier/buyer protocols & tradeoffs
  - » integration of strategic & tactical decision-making
  - » enterprise integration