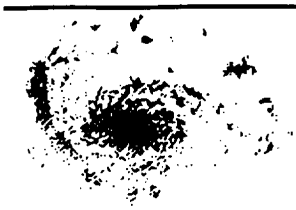
N92-11054



SPIKE

836

SPIKE: AI SCHEDULING TECHNIQUES FOR HUBBLE SPACE TELESCOPE

Mark D. Johnston

GSFC
13 December 1990

Space Telescope Science Institute • Advance Planning Systems Branch
3700 San Martin Drive Baltimore MD 21218

P-1

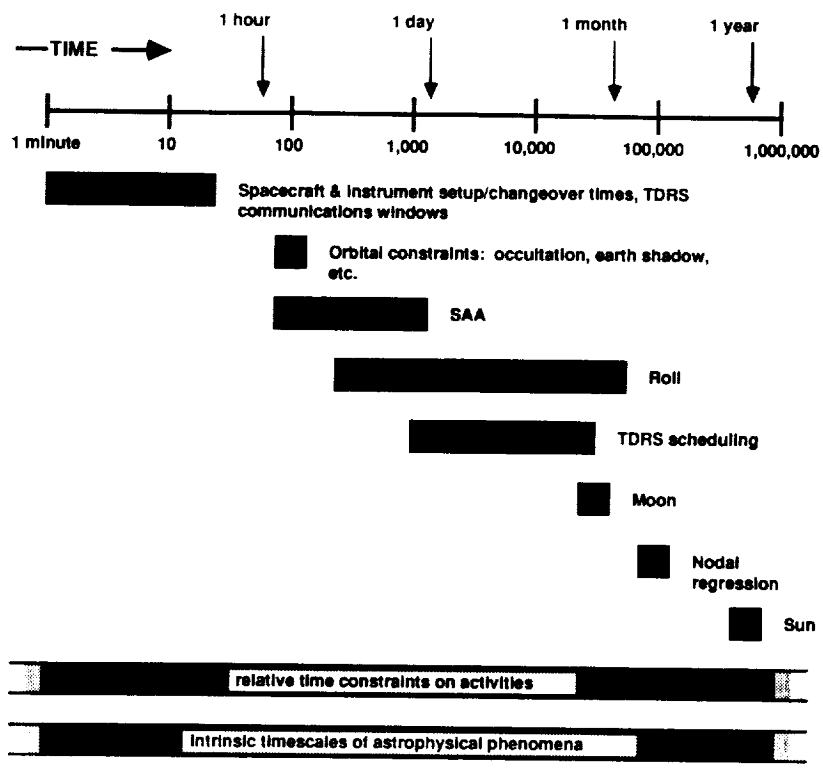
Domain

Hubble Space Telescope (HST) observation scheduling

- HST launched by NASA in April 1990
- 15 year lifetime, low earth orbit (95m period)
- science operations for NASA by Space Telescope Science Institute (STScI) at Johns Hopkins Univ., Baltimore
- HST scheduling is a large problem:
  - ~10,000-30,000 observations/year to be scheduled
  - large number of interacting constraints (~ 10 per observation)
    - operational
    - resource
    - scientific
  - enormous range of constraint timescales (seconds to many months)

P-2

HST Constraint Timescales



P-3

p. 3

HST Scheduling

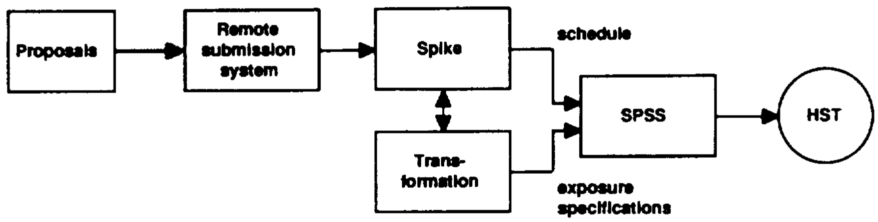
- Predictive scheduling is required by design of spacecraft, ground system
- Pool of observations is intentionally oversubscribed (by about 20%)
- Primary goal is to maximize scientific efficiency of observatory
  - maximize utilization on highest-priority science
  - maximize quality of data taken
- Uncertainty is major problem (orbit, availability of guide stars)
- Spike is a task-oriented scheduler developed by STScI
  - Development started early 1987
  - Current focus: long-range scheduling (one year or more) to resolution of ~days
  - Spike is currently operational and working on flight schedules for period following on-orbit checkout

    Long-term scheduling is on hold pending revision of observing proposals for spherical aberration

P-4

Spike Overview

- Spike draws on two major themes in AI research & applications:
  - constraint satisfaction techniques (search, constraint preprocessing)
  - weight-of-evidence combination for uncertainty reasoning
- Several strategies adopted to decompose problem
- Data flow schematic: from observing proposals to command loads

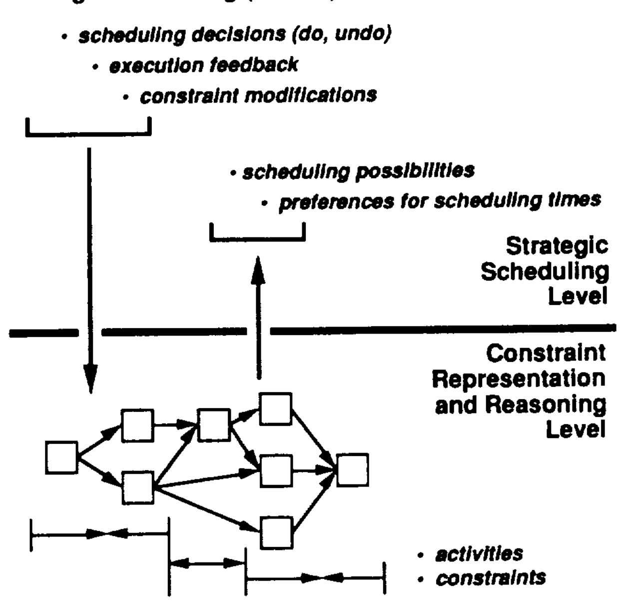


- Spike was designed to support two major modes of use:
  - automatic (offline) scheduling
  - graphical interaction by users, to make scheduling decisions and diagnose scheduling problems

P-5
p. 5

Spike Architecture

- Spike Architecture:
  - low-level constraint representation & propagation
  - higher-level strategic scheduling (search) modules



Constraint Representation & Reasoning

- Temporal constraints and preferences are captured by "suitability functions" based on scheduling expert's assessment:

  the degree of preference for scheduling A_i at t due to constraint α,

  given that A_j, A_k, ... scheduled at t_j, t_k, ..., is $S_i^{\alpha}(t; t_j, t_k, ...)$

- Suitability functions are defined for constraints and derived for tasks
- Projected to functions of time (only) by taking max over possible scheduling times of related activities
- Combined by multiplication: value of 0 means scheduling forbidden, >0 indicates degree of preference
- Combination is formally identical to weight-of-evidence combination in uncertainty reasoning except for special role of overwhelming evidence against scheduling at certain times (S = 0)

P-7

p. 7

Suitability Functions (cont.)

- Task suitabilities are computed by iteration corresponding to:
  node consistency on network of constraints
  + implications of cumulative scheduling decisions
- Value of suitability function informs scheduling agent:
  - times excluded due to strict constraints
  - measure of combined degree of preference due to preference constraints
- For computational efficiency, suitability functions in Spike are represented by piecewise-constant functions of time
  - closed under all important operations
  - no discretization of time or suitability values required
    i.e. no arbitrary limits on time granularity

P-8

Use of Suitability Functions by Scheduling Agent

- **Identify unschedulable activities: $S_i(t) = 0$ for all t**
- **Measure of optimality of schedule: $\prod_i S_i(t_i)$**
- **Measure of potential inherent in partial schedule:**

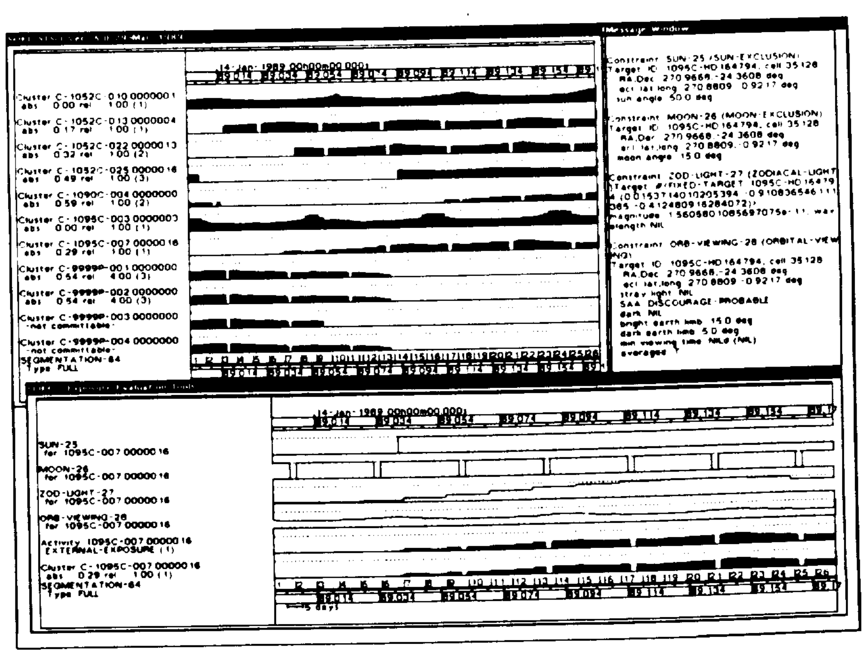
  $\prod_i \max S_i(t)$ indicates best that can be achieved

  - **use to guide search, i.e. explore most promising alternatives first**
- **Explanation: <u>why</u> an activity is unschedulable at t can be determined by examining contributions of constraints to suitability**
  - **guide backtracking at deadends**
  - **give users insight into problem cases: Spike provides graphical display of contributions to strict and preference constraints**

P-9

p. 9

Spike Screen Example



P-10

Advantages of Suitability Function Framework

- Uniform means for simultaneously representing strict (yes/no) and preference constraints
- Framework can represent naturally:
  - trade-offs among preferences
  - uncertainty in predicted scheduling conditions (e.g. high risk ⇒ low suitability)
  - implications of scheduling decisions as they are made
  - implications of task execution as schedule is implemented
- Identify inconsistent constraints & unschedulable activities as soon as feasible
- No times excluded unless in violation of strict constraints or a consequence of prior scheduling decisions
- No bias about future scheduling decisions
- Generally declarative representation ⇒ easy to modify

P-11

p. 11

Limiting Search & Constraint Propagation

Techniques used by Spike:

- Demand-driven constraint propagation
  - i.e. only upon reference to quantities which require constraint consistency
- Time: schedule from coarser to finer time resolution -
  - Formulate constraints to capture essential behavior at relevant timescales
  - Segment scheduling period into sub-intervals, commit, then decompose
- Path consistency -

  For some types of binary constraints it is possible to perform path-consistency before scheduling
  - dramatically speeds constraint propagation during scheduling search
  - identify path-inconsistent constraints before scheduling starts
  - drawback: reduced explanatory capabilities

P-12

Limiting Search (cont.)

- **Activity clustering -**

  Sequence activities into <u>clusters</u> to commit as single entities, considering:

  - **absolute time constraints**
  - **binary relative time constraints in path-consistent form**
  - **heuristics for ordering preferences (e.g. constraint strictness, minimize state change overhead times)**

  – **collapse partially-redundant constraints to their conjunctions**

  – **pull activity constraints up to cluster level and save**

Path Consistency + Activity Clustering ⇒

> order of magnitude reduction in size of problem

P-13
p. 13

Scheduling Search

Several methods provided: all use same underlying constraint representation/propagation mechanism:

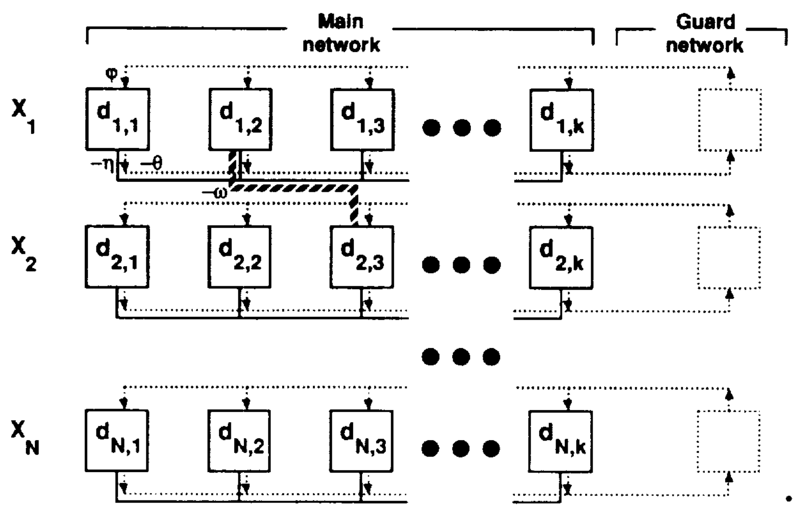
- **Greedy algorithms**
- **Backtracking search**
- **Stochastic ("Neural network")**
- **Repair methods**

Preliminary investigation of re-scheduling algorithms conducted (i.e. where schedule stability is an important goal)

P-14

Stochastic Search

- Developed in collaboration with H.-M. Adorf of Space Telescope - European Coordinating Facility
- Motivated by Hopfield discrete neural network model (but can be formulated as backtracking search using network only for bookkeeping)
- Discretize time: network element represents decision to schedule an activity in a time interval
- Network biases and connections derived directly from suitability functions



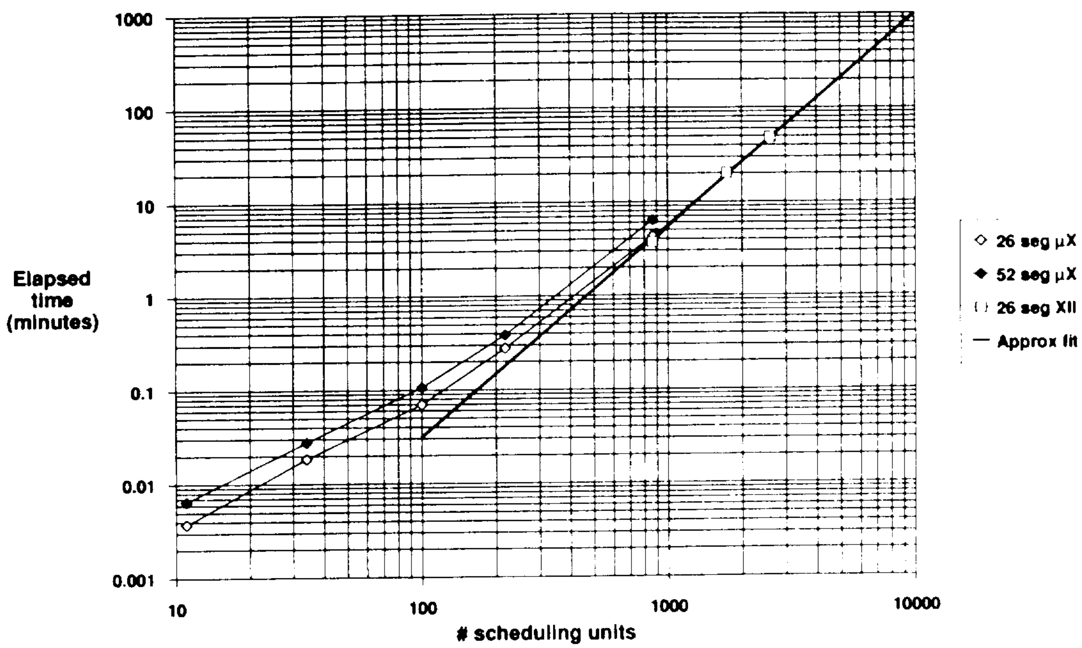
P-15
p. 15

Stochastic Search (cont.)

- Couple to additional networks representing
  - constraint that activity must be scheduled sometime
  - resource (capacity) constraints
- Approach is well-suited for satisficing search where optimization is desired but infeasible
- Interesting characteristics of search:
  - backtracking from deadends and extending partial assignments are simultaneous competing processes
  - tends to maximize overall degree of preference represented by suitability functions
    - i.e. schedules tend towards optimal
  - permits temporary constraint inconsistencies but will not terminate until there are none
  - may not converge (stop and restart)
- By far most effective search strategy in Spike to date
- Performance demonstrated to be adequate for large-scale HST problem

P-16

Neural Network Search Timings



P-17

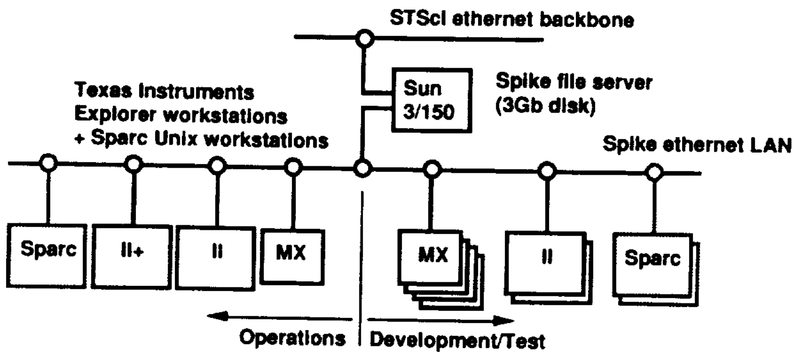
Repair Methods

- Recent work is concentrating on repair methods
  - analysis of neural network operation has isolated several heuristics that explain its success, e.g. min-conflicts
  - theoretical analysis of model problems has identified other heuristics that further improve search performance
- Repair heuristics can be applied in framework that preserves performance of neural network but adds flexibility
- Ideal for reactive re-scheduling
- Machine-learning techniques are being applied to repair methods to "learn" best strategies

P-18

Implementation

- CommonLisp, old Flavors, conversion to CLOS just completed
- CommonWindows for user I/F
- Developed and operated on TI Explorer Lisp machines



- Unix port of core system & user I/F completed December 1989 (using X-windows based CommonWindows); plan for operations migration to Sparcstations over next year

P-19

p. 19

Status

- Spike is in operational use at STScI for scheduling the period following HST instrument checkout and calibration
  - long-term scheduling has been delayed by optics problems
  - Spike is being used for scheduling feasibility checking on shorter timescales
- Use on other problems has been demonstrated:
  - Spike now running at UC Berkeley for scheduling NASA's Extreme Ultraviolet Explorer ('92)
  - MIT plans to use Spike for scheduling X-ray Timing Explorer mission ('94)
- Ongoing work at STScI on performance improvements, repair & rescheduling, short-term scheduling, portable version

P-20

p. 20