More About Planning And Scheduling

CS410/510CS Lecture 9

Bart Massey <bart@cs.pdx.edu>
PSU CS Department
March 8, 2001

Overview

- Pathfinding Search: Aircraft Routing Problem
- Scheduling: Aircraft Assembly Problem
- Extended Scheduling: Cable Assembly Problem
- Planning: Directionality Of Prop. STRIPS
- Planning Domain Reversal
- Determining Planner Direction
- Conclusions

Pathfinding Search: Aircraft Routing Problem

Goal: given

- weather map
- flight model for aircraft

find minimum-fuel-cost route
subject to flight rules

The Aircraft Routing Problem: Weather

Weather map is on 1-degree grid with 10 points per edge, at approx. 25 altitudes. Result is constellation of routes

The Aircraft Routing Problem: Flight Model

Flight model for given plane: high-order polynomials on

- Weight
- Altitude
- Air Pressure and Temperature
- Wind

Very accurate, but computationally expensive

The Great-Circle Heuristic

Obvious heuristic: to get from point A to B, fly "straight line" == great circle

- What altitude to fly at?
- What about winds?

The Admissible Great-Circle Heuristic

Can make great-circle admissible by

- Flying at optimal altitude
- Assuming optimal winds for entire planet over whole route

Obviously can do better: iterated reachability on favorable winds

Two-Pass A*

A^* is optimal, but assumes

- edge cost computation cheap
- search space fits in memory

Alternate approach: back-calculate using cheap approximation to *bound* search space, then use A^* forward

Result is “football” around optimal route

Results and Conclusions

Routes are calculated very quickly (several seconds around the world)

Most good routes near great circle, so no big savings

For strong winds, flies up to 5% better than current USAF tool == huge fuel savings

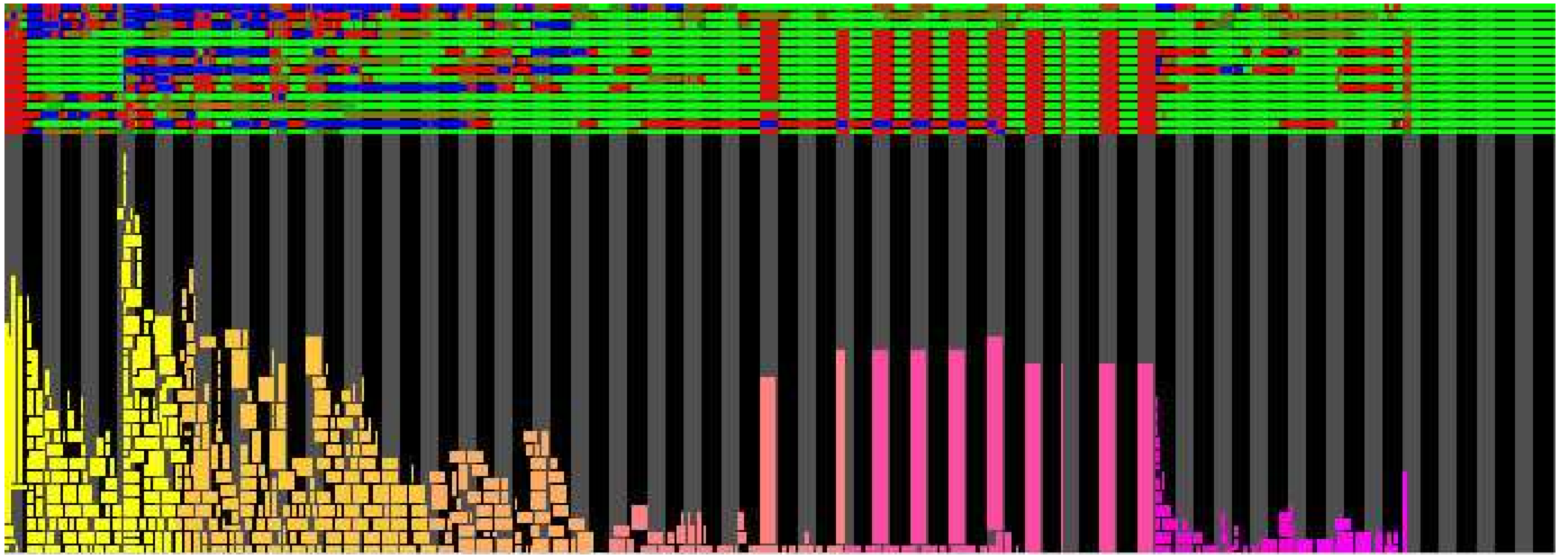
Scheduling: Aircraft Assembly Problem

Talked about problem last week: want to remind, show details

Basic problem: fit 575 tasks into 38 days (LB) or more w/o oversubscribing 17 resources

The Aircraft Assembly Instance

575 tasks, 13 zone resources, 4 labor resources



Priority Scheduling

Last time: precedence-based scheduling

- Select conflicting tasks
- Add precedence

Problem: early commitment == early mistakes

Solution: make precedence apply only when tasks conflict

Doubleback Scheduling

Last time: discussed doubleback idea

- shift right, then left
- repeat until happy or diminishing returns

For this instance, roughly 10 iterations

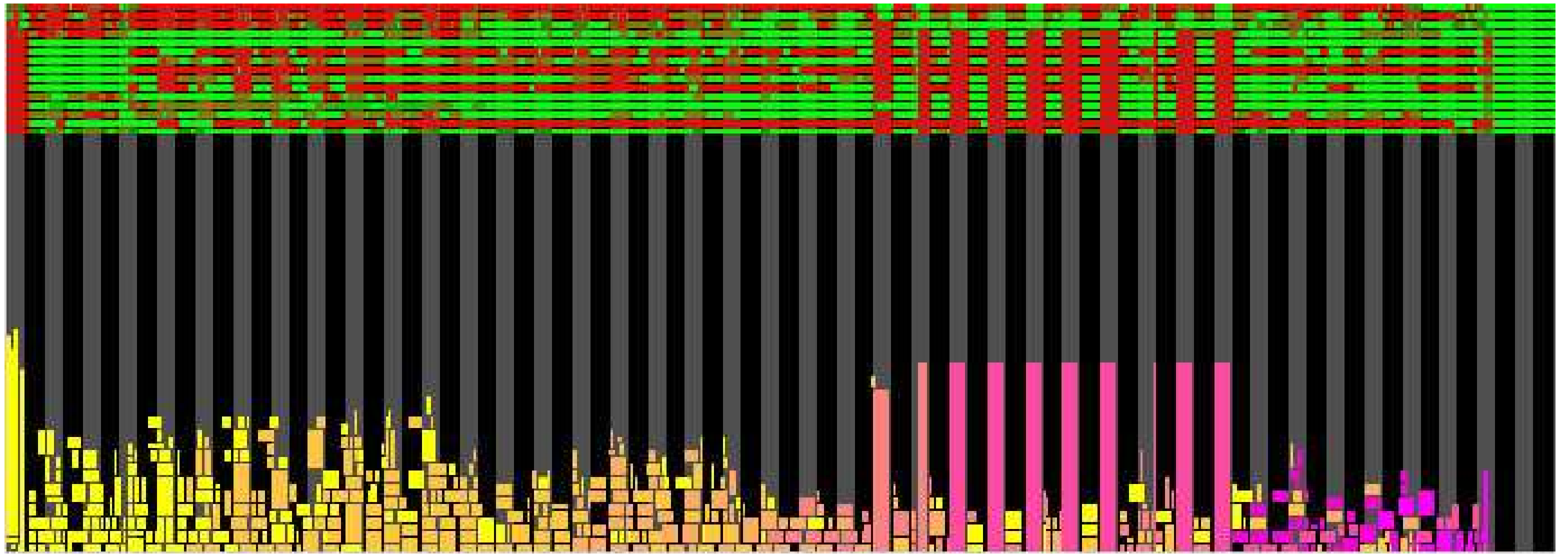
Impact Of LDS On Doubleback

Idea, use Priority Scheduling with LDS, then optimize result with DBO

- Priority Scheduling for “good” initialization
- LDS to spread over search space
- DBO to “clean up” minor heuristic botches

Noise Moves In DBO

Can use straight DBO with slight noise. Here is best (39 day) schedule. Just worse than lower bound



Extended Scheduling: Cable Assembly Problem

Manufacturing problems often lie between scheduling and planning. This problem is scheduling, but with fancy constraints

Novel approach here is response to failure of OR methods. Common practice to

- Observe deficiences in existing methods
- Generalize + algorithmicize sensible solution
- Implement and test

Fiber Optic Cable Assembly

Typical (large) instance: 297 cable bundles, 13 production lines

Features:

-
- Release time, deadline
- Per-machine cost of cable
- Setup cost between cables
- Objective: minimize setup cost + lateness

OR Methods For Assembly Problems

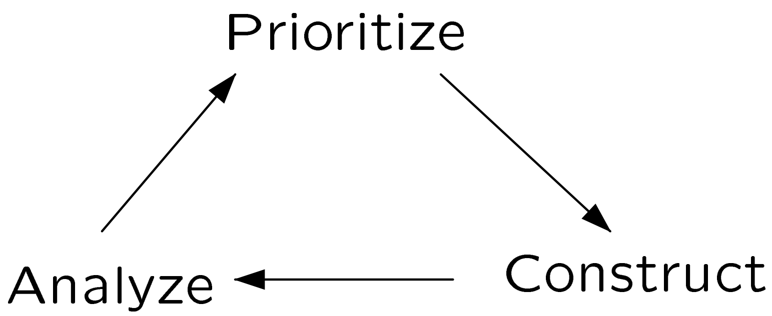
Operations Research approach used

- Linear programming
- BnB search to integerize (make whole cable on 1 machine)

Too slow for large instances, not too good on smaller ones

Squeaky Wheel Optimization

David Joslin at CIRL: use greedy constructor, let order/priority of cables determine schedule



Prioritization In SWO

- Charge bad setups against pair of cables
- Charge late cables
- Sort so worst-placed cables are inserted first in next iteration
- But do not allow big change in priority on one iteration: helps prevent thrashing

Squeaky Wheel On Cable Assembly

Results: dramatically better solution in seconds, solves largest instance. Technique generalizes (e.g. graph coloring)

Planning: Directionality Of Prop. STRIPS

Last time: discussion of Propositional STRIPS (PROPS) planning == action selection + scheduling

Today: closer look at search in PROPS. Focus: search from initial to goal or goal to initial.

- Which (if either) is right?
- What do existing planners do?

More About PROPS

Operators (action types) of form

$$\frac{\textit{clear-a} \quad \textit{clear-c} \quad \textit{on-a-b}}{\textit{clear-b} \quad \textit{on-a-c} \quad \neg\textit{on-a-b} \quad \neg\textit{clear-c}}$$

where top is preconditions, bottom is effects, negated effects are deleted, positive are added

Preconditions, effects are conjunctions of fluents

PROPS Extensions

- Closed World Assumption
- Predicates (or FOL)
- Types (and type predicates)
- Conditional Effects: extra preconditions for additional effects
- Domain Axioms: formulae that are always forced true/false
- Safety Conditions: formulae that effects must not falsify

Planning Search Direction

Usually do not select, schedule actions for “middle” of working plan: hard, counterintuitive. But work from initial or goal?

- Initial: can calculate effect of actions quickly
- Goal: can ensure actions serve purpose (“means-end”) easily

Existing planners use odd spaces, algorithms: may not understand true search direction!

Planning Domain Reversal

Suppose one could

- Swap initial, goal conditions
- Reverse preconditions, effects of operators

to get "bizarro" domain

Then forward bizarro planning == backward planning and backward bizarro planning == forward planning

Reversing An Operator

Exchange preconditions and effects?

$$\frac{\textit{cocoa-in} \quad \textit{water-in}}{\textit{stirred} \quad \textit{spoon-wet}}$$

becomes

$$\frac{\textit{stirred} \quad \textit{spoon-wet}}{\textit{cocoa-in} \quad \textit{water-in}}$$

Fails: no way to express **don't-care** precondition as effect

Don't Care (and Don't Know) Effects

Replace each fluent with "positive" and "negative" versions

$$\frac{cocoa-in^{+} \quad water-in^{+}}{\begin{array}{c} cocoa-in^{+} \quad water-in^{+} \quad stirred^{+} \quad \neg stirred^{-} \\ spoon-wet^{+} \quad \neg spoon-wet^{-} \end{array}}$$

Now reverse to get

$$\frac{cocoa-in^{+} \quad water-in^{+} \quad stirred^{+} \quad spoon-wet^{+}}{\begin{array}{c} cocoa-in^{+} \quad \neg cocoa-in^{-} \quad water-in^{+} \quad \neg water-in^{-} \\ stirred^{+} \quad stirred^{-} \quad spoon-wet^{+} \quad spoon-wet^{-} \end{array}}$$

Really Reversing

Thus, by reversing each fluent using DC/DK effects as needed, can reverse *all* operators. Can also reverse DC goals: become DK initial!

Bizarro planning domain is now constructible:

- No new operators
- Only double number of fluents
- Works fine with existing planners

Reversed Operators and Reversed Plans

Proof idea: mapping domain to bizarro domain, planning, mapping plan back is possible iff legal plan

$$
\begin{tikzcd}
\text{Domain} \arrow[r] \arrow[d] & \text{Plan} \\
\text{Reversed Domain} \arrow[r] & \text{Reversed Plan} \arrow[u]
\end{tikzcd}
$$

Determining Planner Direction

Early planners were theorem provers: direction was not considered

Later planners were forward state space: direction was obvious

Modern planners use

- clever search spaces
- clever search algorithms

Sometimes not obvious what direction. Sometimes inventor guesses wrong!

Search Space Planners

Generalized framework for talking about direction: Search Space Planning. *Approach* is

- Set of actions in plan
- Partial order on actions

Can *propagate* if only one *extension* of approach is possible

One Way Functions

Theory sez: unless P = NP, exist functions H such that

- Computing H is easy (polytime)
- Computing H^{-1} is intractable

Cryptography calls candidates *cryptographic hash functions*

Nondeterministic Boolean Circuits

Boolean circuit is DAG of boolean gates. *Nondeterministic Boolean Circuit* (NBC) gates can produce more than one output for given inputs

Natural match for planning operators

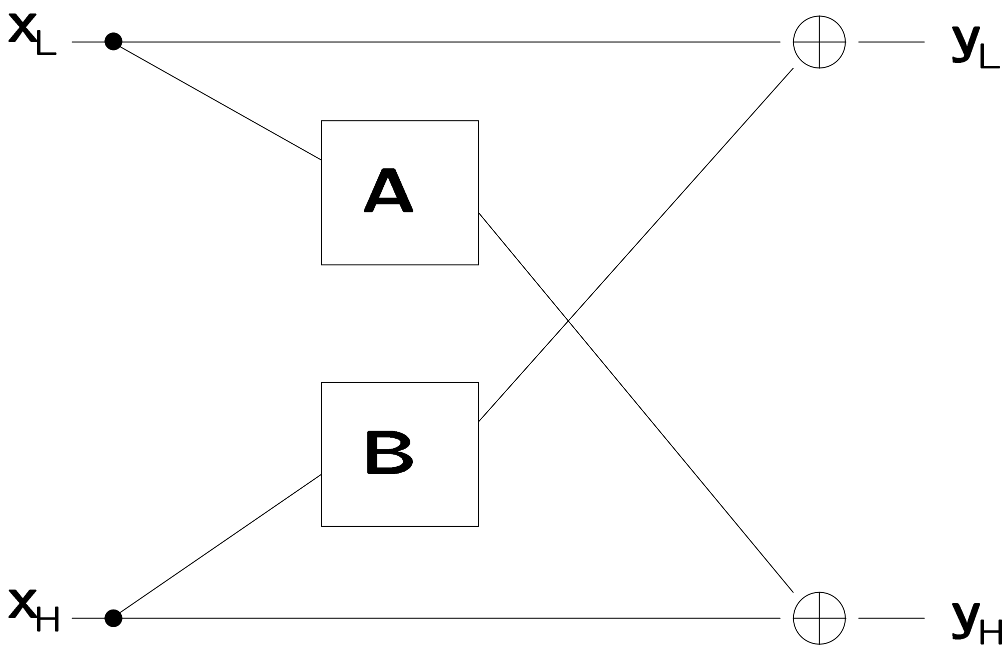
- Preconditions are inputs
- Effects are outputs
- Multiple operators for given preconditions

Planning As Circuit Evaluation

Evaluating NBC $\longrightarrow$ constructing plan. Thus NBC-SAT $\longrightarrow$ plan-existence

- Says planning is hard
- Says can encode one-way function eval as planning!

A Cheap Hash Function



Testbed

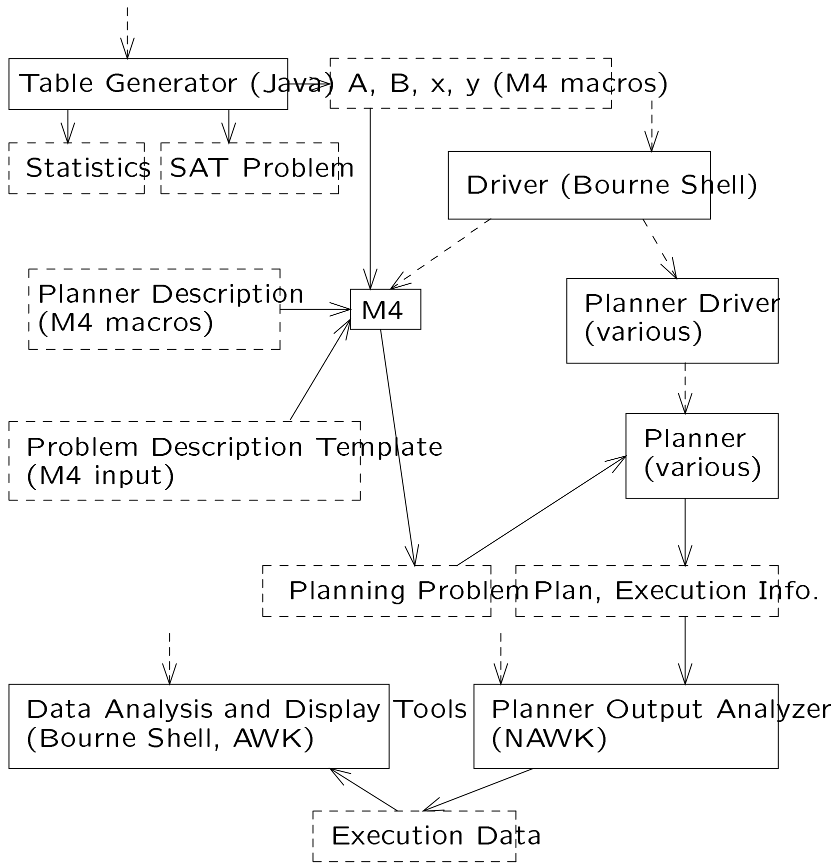


Table Generator (Java)
A, B, x, y (M4 macros)
Statistics
SAT Problem
Driver (Bourne Shell)
Planner Description
(M4 macros)
M4
Planner Driver
(various)
Problem Description Template
(M4 input)
Planner
(various)
Planning Problem
Plan, Execution Info.
Data Analysis and Display Tools
(Bourne Shell, AWK)
Planner Output Analyzer
(NAWK)
Execution Data

Various Planners

- ASP: forward
- O-Plan: backward (?)
- UCPOP: loses
- Graphplan: bidirectional
- Blackbox + Relsat: slightly forward

Conclusions

- In AI, real-world meets theory regularly
- Search direction in planning is interesting
- Don't believe everything you read