Previously

- Resources aren’t free
- Share to reduce costs
- Schedule operations on resources
  - Fixed resources
- Greedy approximation algorithm
- List Scheduling for resource-constrained scheduling
  - Bounds on solution quality

Today

- Time-Constrained Scheduling
  - Force Directed
- Resource-Constrained
  - Branch-and-Bound

Preclass

- Critical Path LB?
- Resources to keep RB < CP?
- Resources to achieve CP?
  - Take poll: 4, 3, 2, 1
- What was trick to achieving?
- Why might List Schedule have a problem with this?

Force-Directed

- **Problem**: how exploit schedule freedom (slack) to minimize instantaneous resources
  - Directly solve time-constrained scheduling
    - (previously only solved indirectly)
  - Minimize resources with timing target
Force-Directed

- Given a node, can schedule anywhere between ASAP and ALAP schedule time
  - Between latest schedule predecessor and ALAP
  - Between ASAP and already scheduled successors
  - Between latest schedule predecessor and earliest schedule successor

- That is: Scheduling node will limit freedom of nodes in path

Force-Directed

- If everything where scheduled, except for the target node, what would we do?:
  - examine resource usage in all timeslots allowed by precedence
  - place in timeslot that has least increase maximum resources
    - Least energy
    - Where the forces are pulling it

Force-Directed Estimate

- Assume a node is uniformly distributed within slack region
  - between earliest and latest possible schedule time
  - In all timesteps between ASAP and ALAP

\[
W(\text{node}) = \frac{1}{\text{slack} + 1}
\]
Single Resource Challenge

Slacks on all nodes

In order to estimate, will need to break each task into fractions. With slack 8, can go in any of 9 slots. With slack 2, can go into any of 3 slots.

Uniform Distribution of Slack

With slack 8, can go in any of 9 slots. With slack 2, can go into any of 3 slots.

Most Constrained Node, Most Used Timeslot
What do to slack of predecessors?

Force-Directed

• Scheduling a node will shift distribution
  – all of scheduled node’s cost goes into one timeslot
  – predecessor/successors may have freedom limited so shift their contributions

• Goal: shift distribution to minimize maximum resource utilization (estimate)
What do to slack of Predecessors?

Must now come before TS = 3.

Slack drops: 2 → 1

\[ W = \frac{1}{1+1} = \frac{1}{2} \]

What do to slack of Successors?

Must now come after TS = 3.

Slack drops: 2 → 1

\[ W = \frac{1}{1+1} = \frac{1}{2} \]

3 4/9

Consequence of assign to TS = 3
What if assign TS=4?

Review TS=2

Review TS=3

Review TS=4
Select TS = 2 to minimize Max weight.
Single Resource Challenge

2 13/18

2 13/18

2 13/18

2 13/18

2 13/18

2 13/18
Force-Directed Algorithm

1. ASAP/ALAP schedule to determine range of times for each node
2. Compute estimated resource usage
3. Select a move
   - For each unscheduled op
     - Evaluate effects of placing in feasible time slots (compute forces)
     - Select move results in minimum cost
     - Repeat until done

Force-Directed Runtime

- Evaluate force of putting in timeslot O(N)
  - Potentially perturbing slack on net prefix/postfix for this node → N
- Each node potentially in T slots: NT
  - T = schedule target
  - N nodes to place: N
  - O(N^2T)
    - Loose bound--don’t get both T slots and N perturbations

Force-Directed Algorithm (from reading)

1. ASAP/ALAP schedule to determine range of times for each node
2. Compute estimated resource usage
3. Select a move
   - For each unscheduled op
     - Evaluate effects of placing in feasible time slots (compute forces)
     - Select move results in minimum cost
     - Repeat until done

Force-Directed Runtime (from reading)

- Evaluate force of putting in timeslot O(N)
  - Potentially perturbing slack on net prefix/postfix for this node → N
- Each selection N nodes to consider: N
- Each node potentially in T slots: NT
  - T = schedule target
  - N nodes to place: N
  - O(N^2T)
    - Loose bound
How Greedy?

- **Class FD:**
  - Greedy selection of what to schedule next
  - Exhaustively consider where might go
- **Reading FD:**
  - Exhaustively consider what to schedule next and where might go
  - Greedy node assignment
    - Never revisit
- **Exhaustive**
  - Don’t commit to an assignment
    - Consider all possible assignments

Branch-and-Bound

(for resource-constrained scheduling)

Brute-Force Scheduling (Exhaustive Search)

- Try all schedules
- Branching/Backtracking Search
- Start w/ nothing scheduled (ready queue)
- At each move (branch) pick:
  - available resource time slot
  - ready task (predecessors completed)
  - schedule task on resource
    - Update ready queue

Example

Branching Search

- Explores entire state space
  - finds optimum schedule
- Exponential work
  - $O(N^{\text{resources} \cdot \text{time-slots}})$
- Many schedules completely uninteresting

Reducing Work

1. Canonicalize “equivalent” schedule configurations
2. Identify “dominating” schedule configurations
3. Prune partial configurations which will lead to worse (or unacceptable results)
“Equivalent” Schedules

- If multiple resources of same type
  - assignment of task to particular resource at a particular timeslot is not distinguishing

Keep track of resource usage by capacity at timeslot.

“Non-Equivalent” Schedules

- Prefixes

Pruning Prefixes

- Keep track of scheduled set
- Recognize when solving same sub-problem
  - Like dynamic programming finding same sub-problems
  - But no guarantee of small number of subproblems
  - Set is power-set so $2^N$
  - ...but not all feasible,
    - so shape of graph may simplify

Dominant Schedules

- A strictly shorter schedule
  - scheduling the same or more tasks
  - will always be superior to the longer schedule

Pruning

- If can establish a particular schedule path will be worse than one we’ve already seen
  - we can discard it w/out further exploration
- In particular:
  - LB=current schedule time + lower_bound_estimate
  - if LB greater than known solution, prune
Pruning Techniques

Establish Lower Bound on schedule time
- Critical Path (ASAP schedule)
- Resource Bound

Alpha-Beta Search

- Generalization
  - keep both upper and lower bound estimates on partial schedule
    - Lower bounds from CP, RB
    - Upper bounds with List Scheduling
  - expand most promising paths
    - (least upper bound, least lower bound)
  - prune based on lower bounds exceeding known upper bound
    - (technique typically used in games/Chess)

Alpha-Beta

- Each scheduling decision will tighten
  - lower/upper bound estimates
- Can choose to expand
  - least current time (breadth first)
  - least lower bound remaining (depth first)
  - least lower bound estimate
  - least upper bound estimate
- Can control greediness
  - weighting lower/upper bound
  - selecting "most promising"

Note

- Aggressive pruning and ordering
  - can sometimes make polynomial time in practice
  - often cannot prove will be polynomial time
  - usually represents problem structure we still need to understand
- Coudert shows scheduling
  - Exact Coloring of Real-Life Graphs is Easy", in Proc. of 34th DAC, Anaheim, CA, June 1997.

Multiple Resources

- Works for multiple resource case
- Computing lower-bounds per resource
  - resource constrained
- Sometimes deal with resource coupling
  - e.g. must have 1 A and 1 B simultaneously or in fixed time slot relation
    - e.g. bus and memory port

Summary

- Resource estimates and refinement
- Branch-and-bound search
  - "equivalent" states
  - dominators
  - estimates/pruning
**Big Ideas:**

- Estimate Resource Usage
- Use dominators to reduce work
- Techniques:
  - Force-Directed
  - Search
    - Branch-and-Bound
    - Alpha-Beta

**Admin**

- Assignment 5 due Thursday
- Reading for Wednesday online
- Office Hours Tuesday