

CS137: Electronic Design Automation

Day 14: March 3, 2004
Scheduling
Heuristics and Approximation



CALTECH CS137 Winter2004 -- DeHon

Today

- Scheduling
 - Force-Directed
 - List-Scheduling
 - Approximation Algorithms

CALTECH CS137 Winter2004 -- DeHon

Last Time

- Resources aren't free
- Share to reduce costs
- Schedule operations on resources
- Greedy not optimal
- Optimal solution with Branch-and-Bound
- Lower Bound Pruning

CALTECH CS137 Winter2004 -- DeHon

This Time

- Heuristic/non-optimal approaches
- Use to solve quickly
- Use to generate upper bounds
 - help pruning in alpha-beta search
- Bounds on “badness” of heuristic

CALTECH CS137 Winter2004 -- DeHon

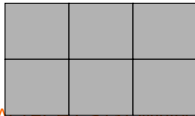
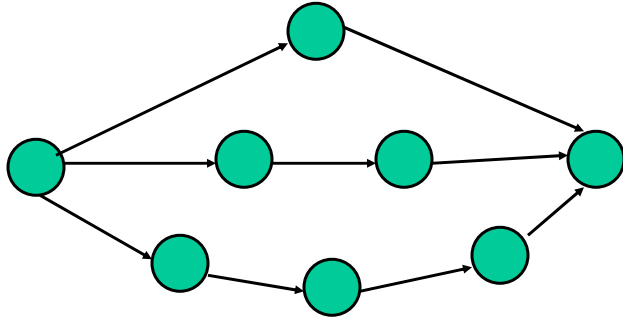
Force-Directed

- **Problem:** how exploit schedule freedom (slack) to minimize instantaneous resources
 - (time constrained)

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
- Scheduling node will limit freedom of nodes in path

Example



CALTECH CS137 Winter2004 -- DeHon

Force-Directed

- If everything were scheduled, **except** for the target node, we would:
 - examine resource usage in all timeslots allowed by precedence
 - place in timeslot which has least increase maximum resources

CALTECH CS137 Winter2004 -- DeHon

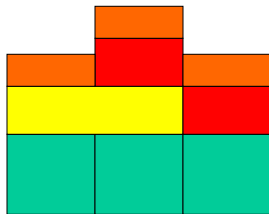
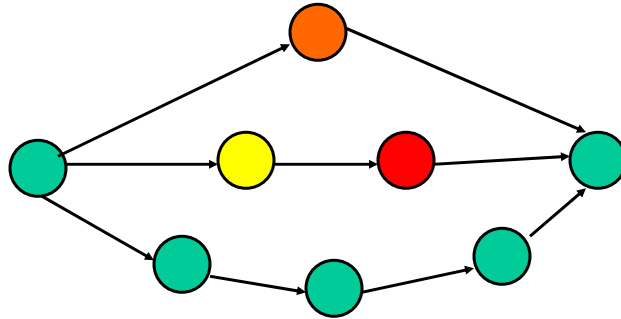
Force-Directed

- **Problem:** don't know resource utilization during scheduling
- **Strategy:** estimate resource utilization

Force-Directed Estimate

- Assume a node is uniformly distributed within slack region
 - between earliest and latest possible schedule time
- Use this estimate to identify most used timeslots

Example



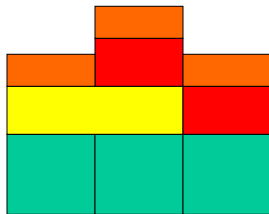
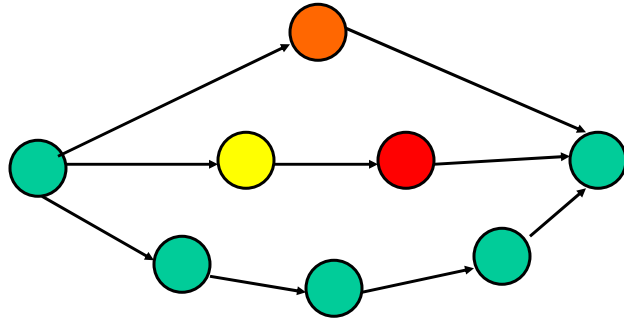
CALTECH CS137 Winter2004 -- DeHon

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
- Want to shift distribution to minimize maximum resource utilization (estimate)

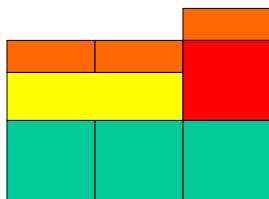
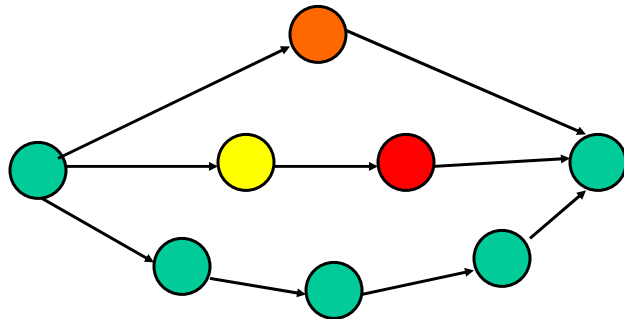
CALTECH CS137 Winter2004 -- DeHon

Example



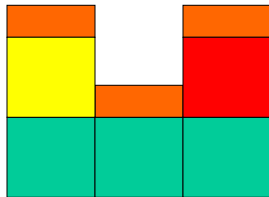
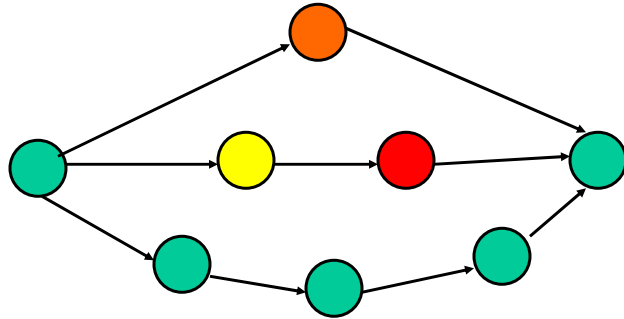
CALTECH CS137 Winter2004 -- DeHon

Example



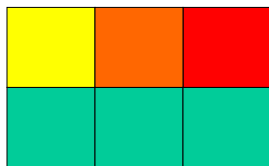
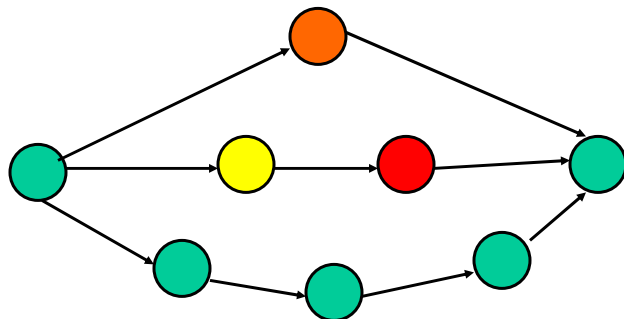
CALTECH CS137 Winter2004 -- DeHon

Example



CALTECH CS137 Winter2004 -- DeHon

Example



CALTECH CS137 Winter2004 -- DeHon

Force-Directed Algorithm

1. ASAP/ALAP schedule to determine range of times for each node
2. Compute estimated resource usage
3. Pick most constrained node
 - Evaluate effects of placing in feasible time slots (compute forces)
 - Place in minimum cost slot and update estimates
 - Repeat until done

CALTECH CS137 Winter2004 -- DeHon

Time

- Evaluate force of putting in timeslot $O(NT)$
 - Potentially perturbing slack on net prefix/postfix for this node $\rightarrow N$
 - Each node potentially in T slots
- Evaluate all timeslots can put in $O(NT^2)$
- N nodes to place
- $O(N^2T^2)$
 - Loose bound--don't get both T slots and N perturbations

CALTECH CS137 Winter2004 -- DeHon

List Scheduling

CALTECH CS137 Winter2004 -- DeHon

List Scheduling (basic algorithm flow)

- Keep a ready list of “available” nodes
 - (one whose predecessors have already been scheduled)
- Pick an unscheduled task and schedule on next available resource
- Put any tasks enabled by this one on ready list

CALTECH CS137 Winter2004 -- DeHon

List Scheduling

- Greedy heuristic
- **Key Question:** How prioritize ready list?
 - What is dominant constraint?
 - least slack (worst critical path)
 - enables work
 - utilize most precious resource
- Last time:
 - saw that no single priority scheme would be optimal

CALTECH CS137 Winter2004 -- DeHon

List Scheduling

- Use for
 - resource constrained
 - time-constrained
 - give resource target and search for minimum resource set
- Fast: $O(N) \rightarrow O(N \log(N))$ depending on prioritization
- Simple, general
- How good?

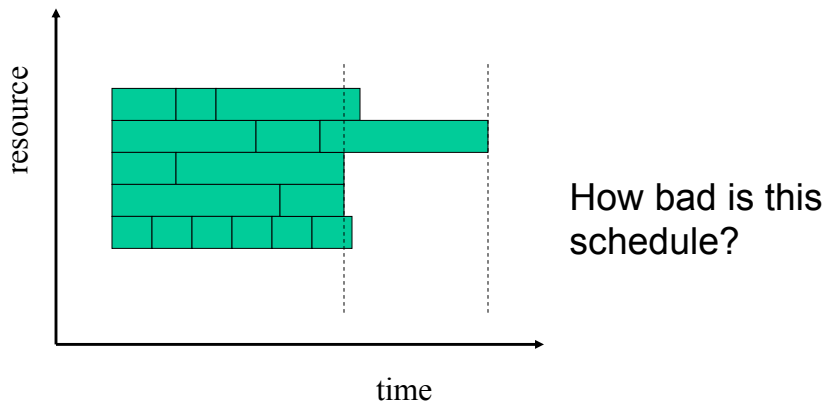
CALTECH CS137 Winter2004 -- DeHon

Approximation

- Can we say how close an algorithm comes to achieving the optimal result?
- Technically:
 - **If** can show
 - $\text{Heuristic}(\text{Prob}) / \text{Optimal}(\text{Prob}) \leq \alpha \quad \forall \text{ prob}$
 - **Then** the Heuristic is an α -approximation

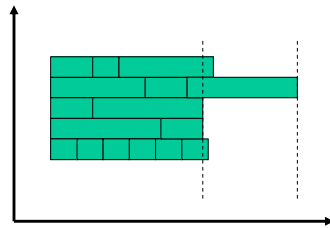
CALTECH CS137 Winter2004 -- DeHon

Scheduled Example Without Precedence



CALTECH CS137 Winter2004 -- DeHon

Observe



- \exists optimal length L
- No idle time up to start of last job to finish
- start time of last job $\leq L$
- last job length $\leq L$
- Total LS length $\leq 2L$
- Algorithm is within factor of 2 of optimum

CALTECH CS137 Winter2004 -- DeHon

Results

- Scheduling of identical parallel machines has a 2-approximation
 - *i.e.* we have a polynomial time algorithm which is guaranteed to achieve a result within a factor of two of the optimal solution.
- In fact, for precedence unconstrained there is a 4/3-approximation
 - (schedule Longest Processing Time first)

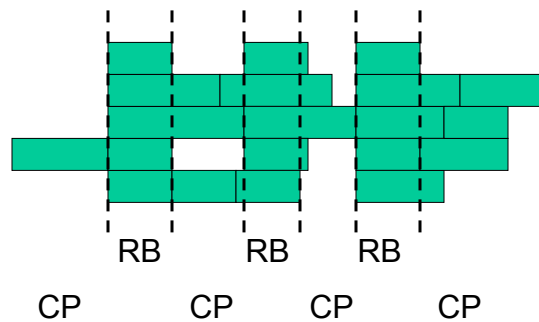
CALTECH CS137 Winter2004 -- DeHon

Recover Precedence

- With precedence we may have idle times, so need to generalize
- Work back from last completed job
 - two cases:
 - entire machine busy
 - some predecessor in critical path is running
- Divide into two sets
 - whole machine busy times
 - critical path chain for this operator

CALTECH CS137 Winter2004 -- DeHon

Precedence



CALTECH CS137 Winter2004 -- DeHon

Precedence Constrained

- Optimal Length $>$ All busy times
 - Optimal Length \geq Resource Bound
 - Resource Bound \geq All busy
- Optimal Length $>$ This Path
 - Optimal Length \geq Critical Path
 - Critical Path \geq This Path
- List Schedule = This path + All busy times
- List Schedule $\leq 2 * (\text{Optimal Length})$

CALTECH CS137 Winter2004 -- DeHon

Conclude

- Scheduling of identical parallel machines with precedence constraints has a 2-approximation.

CALTECH CS137 Winter2004 -- DeHon

Tighten

- LS schedule \leq Critical Path+Resource Bound
- LS schedule \leq $\text{Min}(\text{CP}, \text{RB}) + \text{Max}(\text{CP}, \text{RB})$
- Optimal schedule $\geq \text{Max}(\text{CP}, \text{RB})$
- $\text{LS}/\text{Opt} \leq 1 + \text{Min}(\text{CP}, \text{RB})/\text{Max}(\text{CP}, \text{RB})$

- The more one constraint dominates
 - the closer the approximate solution to optimal
 - ↳ (EEs think about 3DB point in frequency response)

CALTECH CS137 Winter2004 -- DeHon

Tightening

- Example of
 - More information about problem
 - More internal variables
 - ...allow us to state a tighter result
- 2-approx for any graph
 - Since CP may = RB
- Tighter approx as CP and RB diverge

CALTECH CS137 Winter2004 -- DeHon

Multiple Resource

- Previous result for homogeneous functional units
- For heterogeneous resources:
 - also a 2-approximation
 - Lenstra+Shmoys+Tardos, Math. Programming v46p259
 - (not online, no precedence constraints)

CALTECH CS137 Winter2004 -- DeHon

Bounds

- Precedence case, Identical machines
 - no polynomial approximation algorithm can achieve better than $4/3$ bound
 - (unless $P=NP$)
- Heterogeneous machines (no precedence)
 - no polynomial approximation algorithm can achieve better than $3/2$ bound

CALTECH CS137 Winter2004 -- DeHon

Other Approaches

- Graph Coloring
 - Viable with pruning
- ILP
- Annealing

CALTECH CS137 Winter2004 -- DeHon

Summary

- Heuristic approaches to schedule
 - Force-directed
 - List Scheduling
- We can, sometimes, bound the “badness” of a heuristic
 - get a tighter result based on gross properties of the problem
 - approximation algorithms often a viable alternative to finding optimum
 - play role in knowing “goodness” of solution

CALTECH CS137 Winter2004 -- DeHon

Admin

- Assignment #4 due Friday
- Gerez for Monday
 - (handed out last time)
- Pathfinder for Wednesday
 - Grab online

CALTECH CS137 Winter2004 -- DeHon

Today's Big Ideas:

- Exploit freedom in problem to reduce costs
 - (slack in schedules)
- Technique: estimation/refinement
- Use dominating effects
 - (constrained resources)
 - the more an effect dominates, the “easier” the problem
- Technique: Approximation

CALTECH CS137 Winter2004 -- DeHon