Schedule length bounds for optimal task scheduling

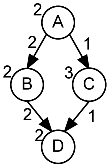
Sarad Venugopalan, **Oliver Sinnen**

PARALLEL AND RECONFIGURABLE COMPUTING GROUP

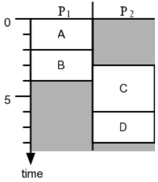
Department of Electrical and Computer Engineering
University of Auckland, New Zealand

Task scheduling with communication delays

Scheduling task graphs with communication delays on homogeneous processors

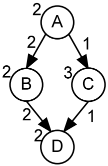


$\Rightarrow$

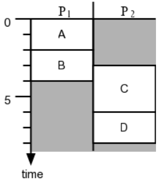


Task scheduling with communication delays

Scheduling task graphs with communication delays on homogeneous processors



$\Rightarrow$



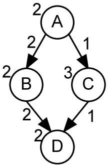
$P|prec, c_{ij}|C_{max}$

- Traditional and general problem
- Strong NP-hard

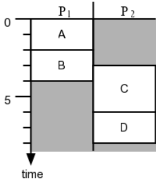
$\Rightarrow$ Heuristics, most popular is list scheduling

Task scheduling with communication delays

Scheduling task graphs with communication delays on homogeneous processors



$\Rightarrow$



$P|prec, c_{ij}|C_{max}$

- Traditional and general problem
- Strong NP-hard

$\Rightarrow$ Heuristics, most popular is list scheduling

But here,

$\Rightarrow$ Optimal solver, based on state space search

$\Rightarrow$ Today solver algorithms that work with limited memory

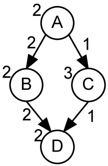
Content

1. Scheduling problem
2. State space search
3. Limited memory searches
4. Lower bounds
   - Destructive lower bound
   - Bounds for certain graph structures
5. Evaluation

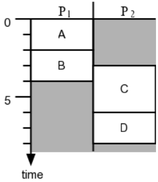
Content

Scheduling problem

Finding start time and processor allocation for every task



$\Rightarrow$

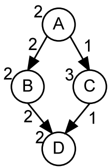


- t_i : start time of task i
- p_i : processor of task i

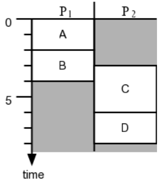
Given by task graph $G = (V, E)$

- L_i : execution time of task i
  - weight of node
- γ_{ij} : remote communication cost between tasks i and j
  - weight of edge

Constraints



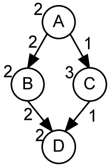
$\Rightarrow$



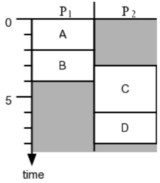
Processor constraint

$$p_i = p_j \Rightarrow \begin{cases} & t_i + L_i \leq t_j \\ \text{or} & t_j + L_j \leq t_i \end{cases}$$

Constraints



$\Rightarrow$



Processor constraint

$$p_i = p_j \Rightarrow \begin{cases} & t_i + L_i \leq t_j \\ \text{or} & t_j + L_j \leq t_i \end{cases}$$

Precedence constraint

For each edge e_{ij} of E

$$t_j \geq t_i + L_i + \begin{cases} 0 & \text{if } p_i = p_j \\ \gamma_{ij} & \text{otherwise} \end{cases}$$

Content

1. Scheduling problem
2. State space search
3. Limited memory searches
4. Lower bounds
   - Destructive lower bound
   - Bounds for certain graph structures
5. Evaluation

Optimal solution techniques

- Mixed Integer Linear Programming – Venugopalan, Sinnen, IEEE TPDS 2014

Optimal solution techniques

- Mixed Integer Linear Programming – Venugopalan, Sinnen, IEEE TPDS 2014
- State Space Search
  - Exhaustive search through all possible solutions
  - Every state (node) s represents partial solution
  - Combinatorial problems $\Rightarrow$ search tree
  - Deeper nodes are more complete solutions

Optimal solution techniques

- Mixed Integer Linear Programming – Venugopalan, Sinnen, IEEE TPDS 2014
- State Space Search
  - Exhaustive search through all possible solutions
  - Every state (node) s represents partial solution
  - Combinatorial problems $\Rightarrow$ search tree
  - Deeper nodes are more complete solutions
- Search techniques
  - A* – great performance, but memory problem !
  - IDA*, Branch and Bound – Limited memory search techniques

Solution space for scheduling problem

Essentially: list scheduling, trying out all task orders and all processor allocations

- State: partial schedule
- Initial state: empty schedule
- Cost function $f(s)$: underestimate of makespan for complete schedule based on s

Solution space for scheduling problem

Essentially: list scheduling, trying out all task orders and all processor allocations

- State: partial schedule
- Initial state: empty schedule
- Cost function $f(s)$: underestimate of makespan for complete schedule based on s

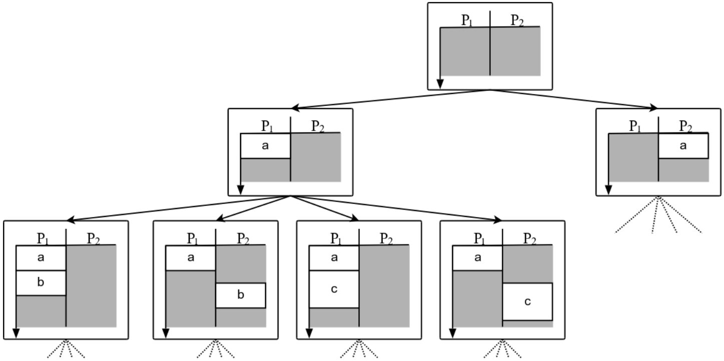
Expansion

- Given state s, let **free**(s) be free tasks

for all $i \in$ **free**(s) **do**
  for all $P \in \mathbf{P}$ **do**
    Create new state: i scheduled on P as early as possible

Solution tree

- Task graph on two processors



Cost function $f(s)$

Three components

- Perfect load balance plus current idle time

$$f_{idle-time}(s) = \frac{\sum_{i \in \mathbf{V}} L_i + idle(s)}{|\mathbf{P}|}$$

Cost function $f(s)$

Three components

- Perfect load balance plus current idle time

$$f_{idle-time}(s) = \frac{\sum_{i \in \mathbf{V}} L_i + idle(s)}{|\mathbf{P}|}$$

- Max (start time of scheduled tasks plus their bottom level)

$$f_{bl}(s) = \max_{i \in s}\{t_i + bl_w(i)\}$$

Cost function $f(s)$

Three components

- Perfect load balance plus current idle time

$$f_{idle-time}(s) = \frac{\sum_{i \in \mathbf{V}} L_i + idle(s)}{|\mathbf{P}|}$$

- Max (start time of scheduled tasks plus their bottom level)

$$f_{bl}(s) = \max_{i \in s}\{t_i + bl_w(i)\}$$

- Unscheduled tasks: Data-Ready-Time plus their bottom levels

$$f_{DRT}(s) = \max_{i \in \mathbf{free}(s)}\{t_{dr}(i) + bl_w(i)\}$$

Cost function $f(s)$

Three components

- Perfect load balance plus current idle time

$$f_{idle-time}(s) = \frac{\sum_{i \in \mathbf{V}} L_i + idle(s)}{|\mathbf{P}|}$$

- Max (start time of scheduled tasks plus their bottom level)

$$f_{bl}(s) = \max_{i \in s}\{t_i + bl_w(i)\}$$

- Unscheduled tasks: Data-Ready-Time plus their bottom levels

$$f_{DRT}(s) = \max_{i \in \mathbf{free}(s)}\{t_{dr}(i) + bl_w(i)\}$$

Complete $f(s)$ function:

$$f(s) = \max\{f_{idle-time}(s), f_{bl}(s), f_{DRT}(s)\}$$

Content

Branch and Bound

- Branch and Bound – can mean many things
- Usual meaning: DFS Branch and Bound

Branch and Bound

- Branch and Bound – can mean many things
- Usual meaning: DFS Branch and Bound

B & B

$B \leftarrow upperBound$
DFS on state space (depth until $f(s) \geq B$):
if complete solution s_c found & $f(s_c) < B$ **then**
  $B \leftarrow f(s_c)$

Branch and Bound

- Branch and Bound – can mean many things
- Usual meaning: DFS Branch and Bound

B & B

$B \leftarrow upperBound$
DFS on state space (depth until $f(s) \geq B$):
if complete solution s_c found & $f(s_c) < B$ **then**
  $B \leftarrow f(s_c)$

- Memory required is $O(|V|P)$
- Benefits from tight upper bounds for initial B

IDA*

- Iterative Deepening A* (IDA*)
- Uses threshold
  - Depth limited by threshold: if $f(s) > \textit{threshold}$ do not descend further

IDA*

- Iterative Deepening A* (IDA*)
- Uses threshold
  - Depth limited by threshold: if $f(s) > \textit{threshold}$ do not descend further

IDA*

$T \leftarrow \textit{lowerBound}$
while no complete solution **do**
  DFS on state space (depth until $f(s) > T$)
  if complete solution found **then**
    Solution is optimal
  else
    Increase T to smallest $f(s) > T$ that was found

IDA*

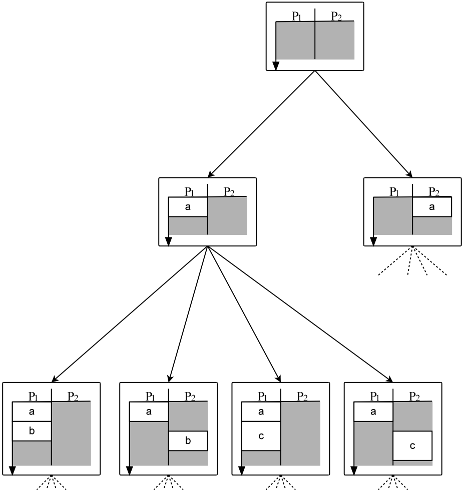
- Iterative Deepening A* (IDA*)
- Uses threshold
  - Depth limited by threshold: if $f(s) > \textit{threshold}$ do not descend further

IDA*

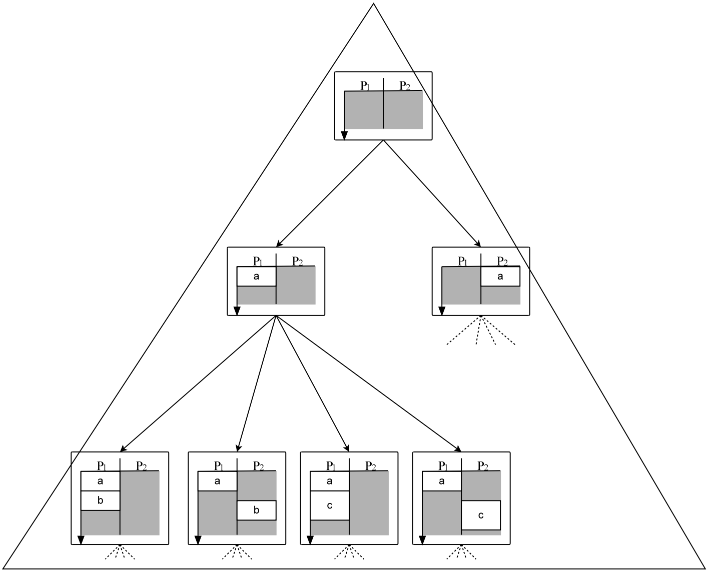
$T \leftarrow \textit{lowerBound}$
while no complete solution **do**
  DFS on state space (depth until $f(s) > T$)
  if complete solution found **then**
    Solution is optimal
  else
    Increase T to smallest $f(s) > T$ that was found

- Memory required is $O(|V|P)$
- Benefits from tight lower bounds initial threshold T

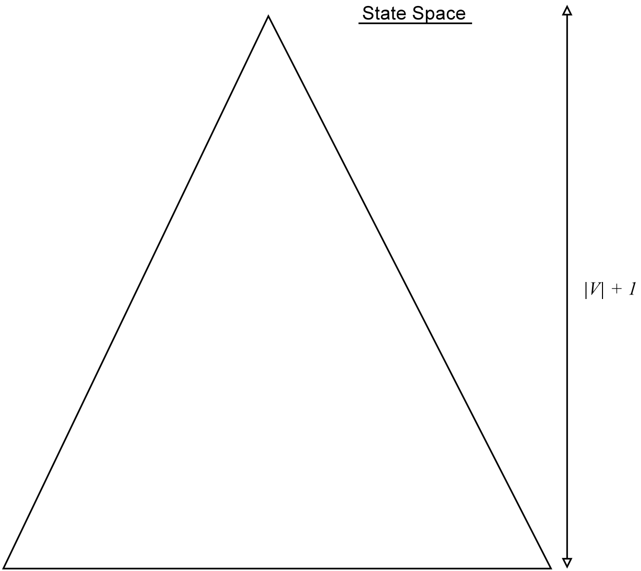
Comparison of state space search techniques



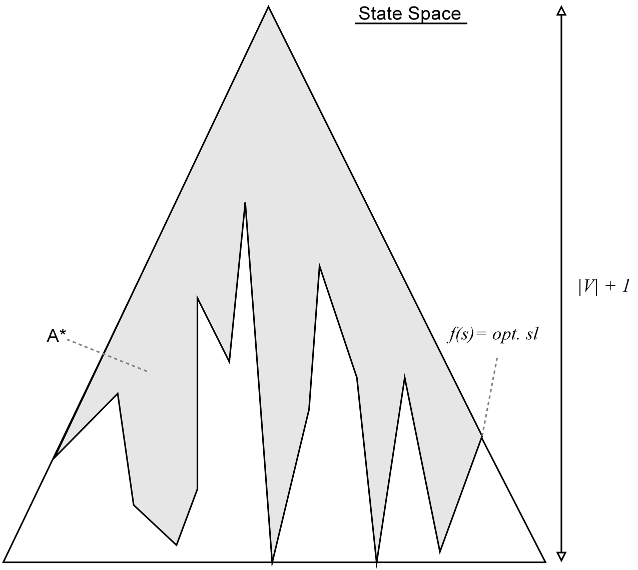
Comparison of state space search techniques



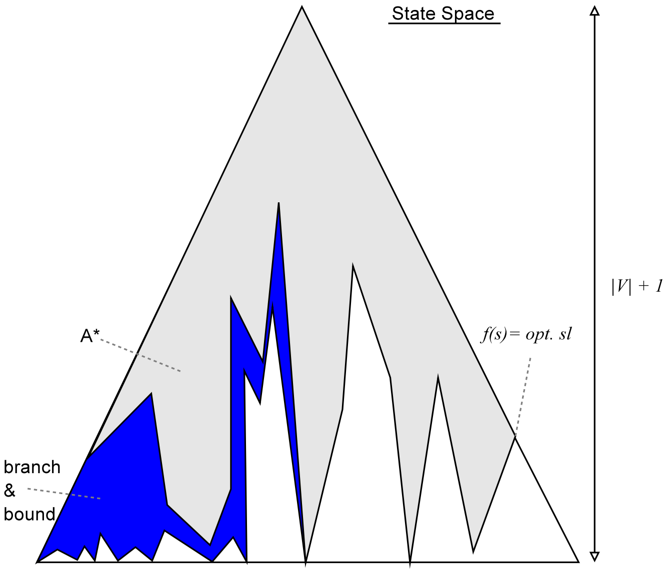
Comparison of state space search techniques



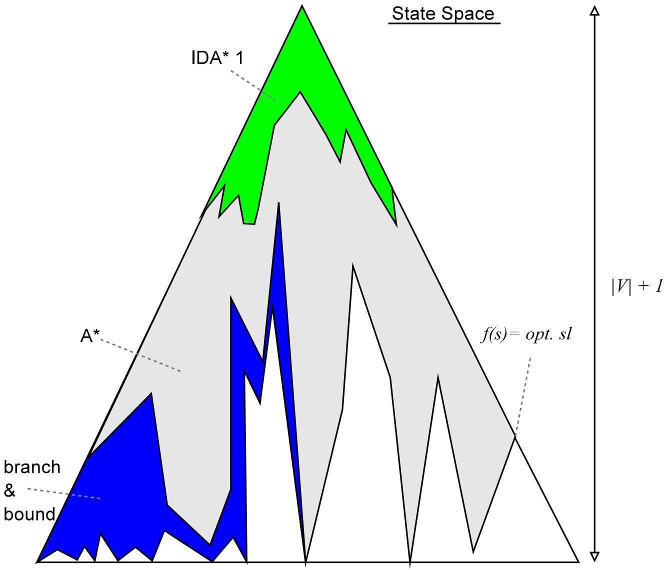
Comparison of state space search techniques



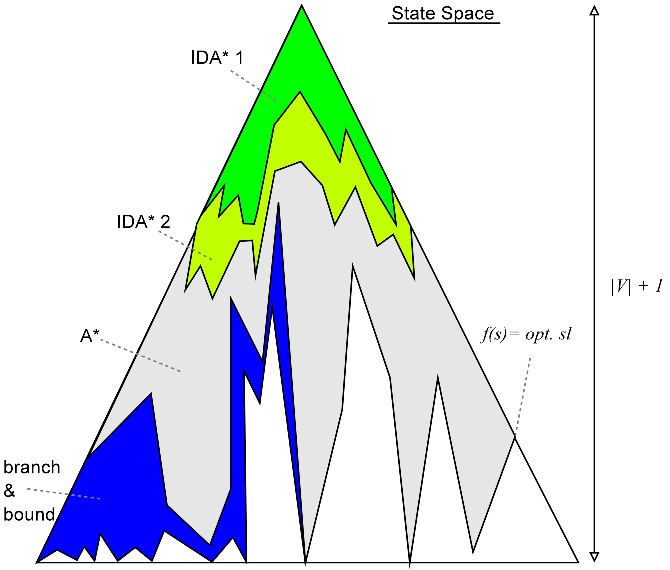
Comparison of state space search techniques



Comparison of state space search techniques



Comparison of state space search techniques



Content

1. Scheduling problem
2. State space search
3. Limited memory searches
4. Lower bounds
   - Destructive lower bound
   - Bounds for certain graph structures
5. Evaluation

Lower bounds – general

Lower bound for any graph

- Critical path length (without communication costs)
  - $sl \geq \sum_{i=1}^{|V|} L_i$

Lower bounds – general

Lower bound for any graph

- Critical path length (without communication costs)
  - $sl \geq \sum_{i=1}^{|V|} L_i$
- Perfect load balance (sum of all task weights divided by number of processors)
  - $sl \geq \max_{i \in V}\{bl(i)\}$

Lower bounds – general

Lower bound for any graph

- Critical path length (without communication costs)
  - $sl \geq \sum_{i=1}^{|V|} L_i$
- Perfect load balance (sum of all task weights divided by number of processors)
  - $sl \geq \max_{i \in V}\{bl(i)\}$

Often not very close (structure and communication costs)

$\Rightarrow$ Improve through ILP constraints and for certain graph types

Content

Destructive lower bound

- Using ILP constraints to improve lower bound (not to solve scheduling problem)

Compute destructive lower bound

Use ILP formulation (plus additional constraints)
Add constraint $\forall t_i \in V, t_i + L_i \leq dlb$
Binary search in $dlb = lowerBound$ to $upperBound$:
Test for constraint violation
if constraints violated **then**
  $lowerBound \leftarrow dlb$
else
  $upperBound \leftarrow dlb$
Repeat until $lowerBound = upperBound$

- Final *lowerBound* is new lower bound on schedule length
  - Note, that *upperBound* is non-conclusive

ILP formulation

min	W	MinMax
$\forall i \in V$	$t_i + L_i \leq W$	
$\forall i \neq j \in V$	$\sigma_{ij} + \sigma_{ji} + \epsilon_{ij} + \epsilon_{ji} \geq 1$	Overlap
$\forall i \neq j \in V$	$\sigma_{ij} + \sigma_{ji} \leq 1$	
$\forall i \neq j \in V$	$\epsilon_{ij} + \epsilon_{ji} \leq 1$	
$\forall j \in V \ : \ i \in \delta^-(j)$	$\sigma_{ij} = 1$	Edge
$\forall j \in V \ : \ i \in \delta^-(j)$	$p_j - p_i \leq \epsilon_{ij}\|P\|$	Processor
$\forall j \in V \ : \ i \in \delta^-(j)$	$p_i - p_j \leq \epsilon_{ji}\|P\|$	
$\forall i \neq j \in V$	$p_j - p_i - 1 - (\epsilon_{ij} - 1)\|P\| \geq 0$	
$\forall i \neq j \in V$	$t_j - t_i - L_i - (\sigma_{ij} - 1)W_{max} \geq 0$	Precedence
$\forall j \in V \ : \ i \in \delta^-(j)$	$t_i + L_i + \gamma_{ij}(\epsilon_{ij} + \epsilon_{ji}) \leq t_j$	

Added constraints

- Adding constraints that make check for constraint violation faster
  - But not solving ILP faster !

Added constraints

- Adding constraints that make check for constraint violation faster
  - But not solving ILP faster !
- Level constraints
  - $t_i \geq tl(i)$ (top level)
  - $t_i \leq W - bl(i)$ (bottom level)

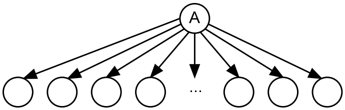
Added constraints

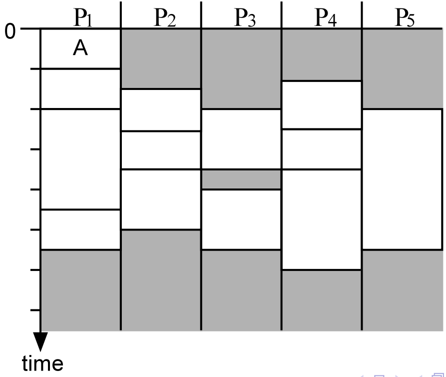
- Adding constraints that make check for constraint violation faster
  - But not solving ILP faster !
- Level constraints
  - $t_i \geq tl(i)$ (top level)
  - $t_i \leq W - bl(i)$ (bottom level)
- Transitive constraints
  - If task i before task j and j before k, then i before k
  - $\epsilon_{ij} + \epsilon_{jk} \geq \epsilon_{ik}$

Content

Lower bound for fork

Example schedule on 5 processors

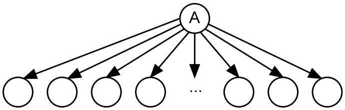


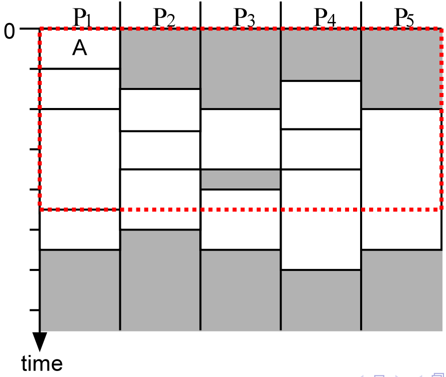


Lower bound for fork

Example schedule on 5 processors

- Red: perfect load balancing

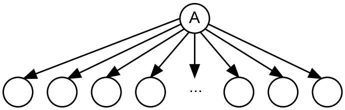


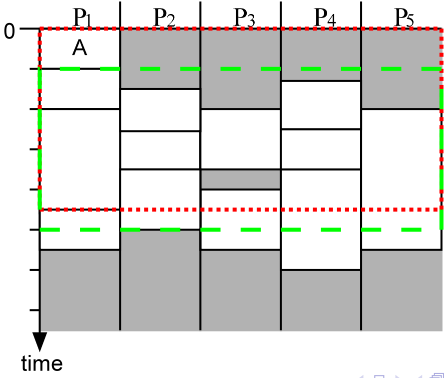


Lower bound for fork

Example schedule on 5 processors

- Red: perfect load balancing
- Green: root task + perfect load balancing

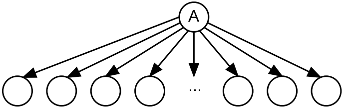


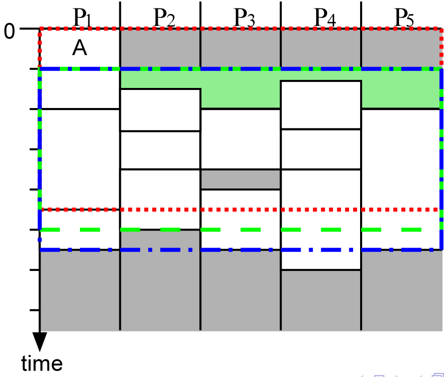


Lower bound for fork

Example schedule on 5 processors

- Red: perfect load balancing
- Green: root task + perfect load balancing
- Blue: root task + perfect load balancing + min. communication cost

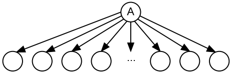


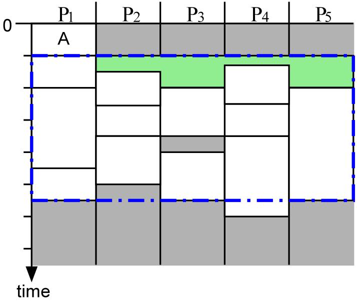


Lower bound for fork

$$LB_F = L_{root} + \min_{1 \leq j \leq |P|} \left\{ \frac{\sum_{i=1}^{|V|} L_i - L_{root} + \sum_{k=1}^{j-1} SCC_k}{j} \right\}$$

where SCC are the smallest incoming communication costs in non-decreasing order

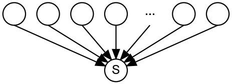


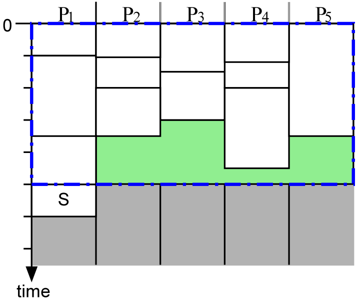


Lower bound for join

$$LB_J = L_{sink} + \min_{1 \leq j \leq |P|} \left\{ \frac{\sum_{i=1}^{|V|} L_i - L_{sink} + \sum_{k=1}^{j-1} SCC_k}{j} \right\}$$

where SCC are the smallest outgoing communication costs in non-decreasing order

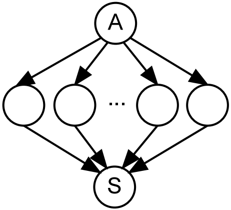


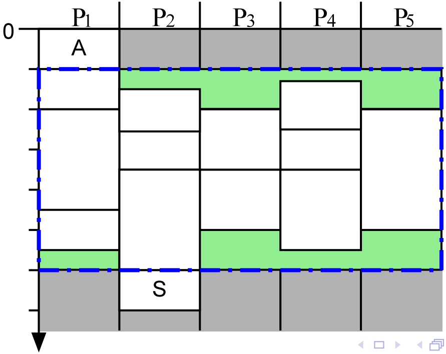


Lower bound for fork-join

$$LB_{FJ} = L_{root} + L_{sink} + \min_{1 \le j \le |P|} \left\{ \frac{\sum_{i=1}^{|V|} L_i - L_{root} - L_{sink} + \sum_{k=1}^{j-1} SCC_k^F + SCC_k^J}{j} \right\}$$

where SCC are the smallest outgoing communication costs in non-decreasing order





Content

Evaluation

- Set of 207 graphs, different structures and sizes

Graph Structure	$n = 10$	$n = 21$	$n = 30$	Total
Fork-Join	4	4	4	12
Fork	4	4	4	12
Independent	1	1	1	3
InTree	8	8	8	24
Join	4	4	4	12
OutTree	8	8	8	24
Pipeline	4	4	4	12
Random	16	16	16	48
Series-Parallel	16	16	16	48
Stencil	4	4	4	12

Improvement in tightness of bound – Destructive

- Count of improved lower bound for the 207 graph database

$p = 2$	$p = 4$	$p = 8$	$p = 16$
59	122	162	166

- Quality in improvement in the lower bound by using destructive lower bounds

	$p = 2$	$p = 4$	$p = 8$	$p = 16$
considered graphs	49	72	88	99
average normalised improvement $sl_{opt} - lb$	41.18%	52.37%	58.87%	56.34%

Improvement in tightness of bound – Structure LB

- Count of graphs with improved lower bound (out of 12 each)

	$p = 2$	$p = 4$	$p = 8$	$p = 16$
fork	12	12	11	10
join	12	12	10	7
fork-join	11	12	10	6

- Quality in improvement of bound by using structure lower bounds

	$p = 2$	$p = 4$	$p = 8$	$p = 16$
considered graphs	5	10	13	9
average normalised improvement $sl_{opt} - lb$	84.96%	72.27%	57.29%	50.65%

Bound impact on IDA*

- Speedup on IDA* (no pruning) through Lower Bound improvements

Graph	n	(LB, LB_{Prop})	Time saved ($p = 2$)	(LB, LB_{Prop})	Time saved ($p = 4$)
random	10	(23,29)	1s	(22,26)	2m:54s
fork	10	(38,45)	51m:44s	(19,31)	52m:22s
join	10	(30,37)	5h:50m	(15,26)	>12h
fork-join	10	(435,494)	1h:38m:49s	(257,308)	>12h

Comparison IDA* and B&B

- What is better? IDA* or B&B?

Comparison IDA* and B&B

- What is better? IDA* or B&B?
- Runtime limit of 1 minute
- Table shows number of obtained optimal schedules within time limit (out of 207)

Number of Processors	Branch and Bound	IDA*
2	93	93
4	73	70
8	69	69
16	62	69

Conclusion

Two new optimal solvers for task scheduling:

- IDA*
- Branch and bound
- Do not run out of memory
- Good bounds on schedule length significantly improve performance
- Proposed mechanisms to improve bounds

Future

- Use IDA* and B&B for gap calculation
- Further pruning techniques
- Extensive comparison between approaches
- Parallelisation