Investigation of some Stochastic Scheduling Problems

Master's Thesis in Computer Science

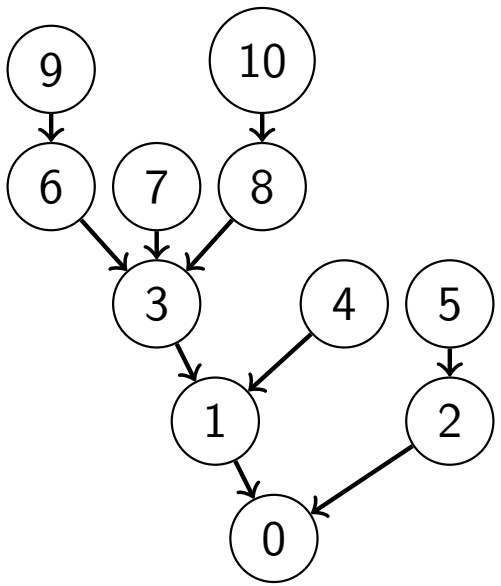
Philipp Müller

Technische Universität München

November 20, 2013

Problem statement

- Set of tasks
- Task times exponentially distributed with parameter 1
- Dependencies form intree structure
- Non-preemptive scheduling
- Goal: Minimize total expected make span



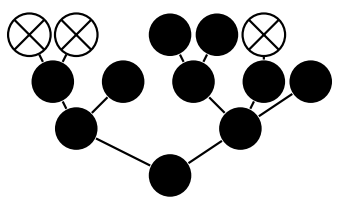
Schedules and snapshots

Schedules

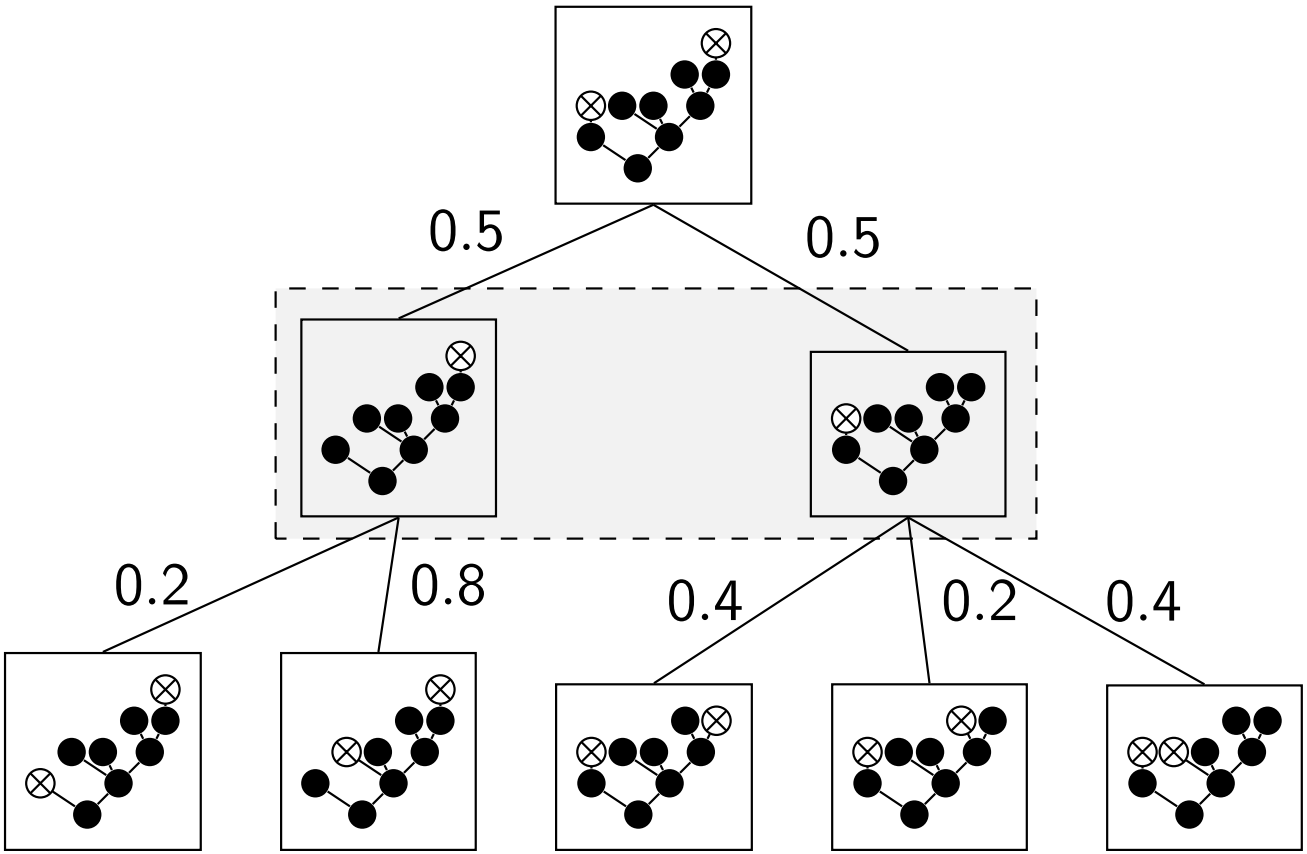
- Describes the order of tasks
- Our scenario: non-deterministic and non-preemptive
- Scheduling *strategy* influences expected make span

Snapshots

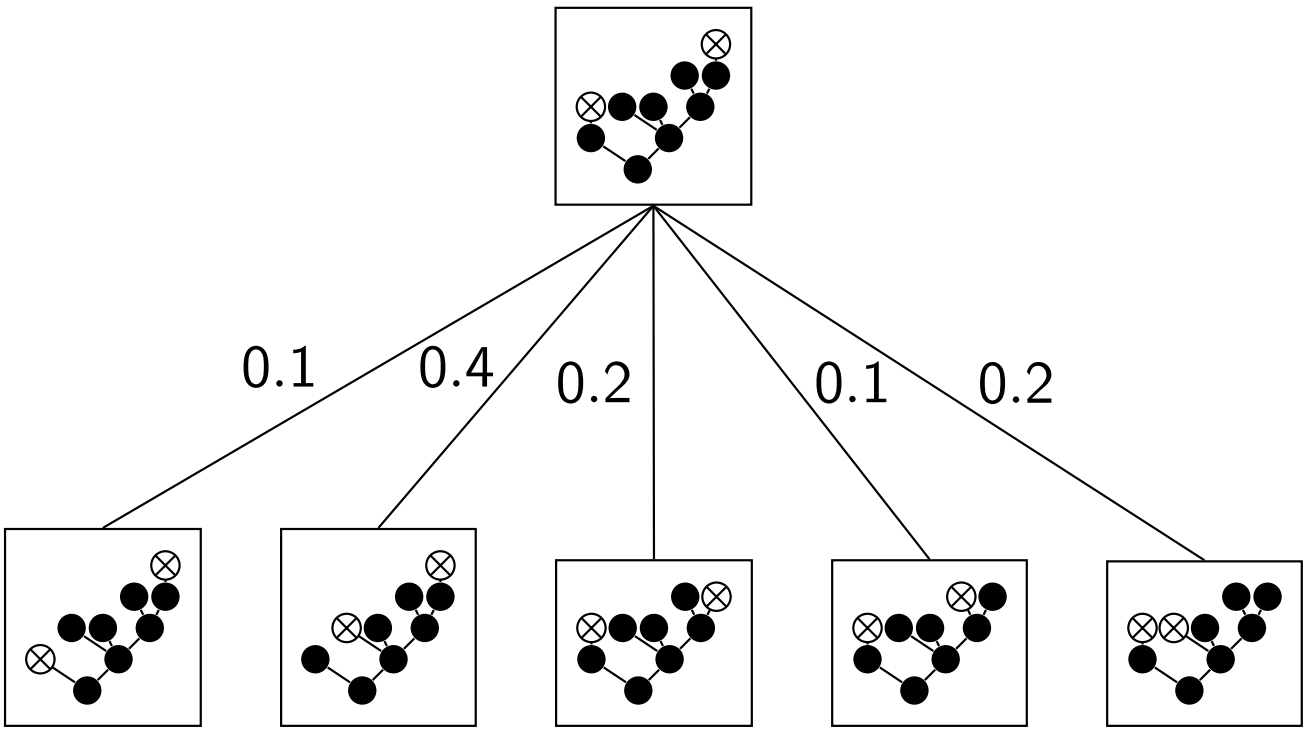
- States of a schedule
- Consist of current intree and the set of scheduled tasks



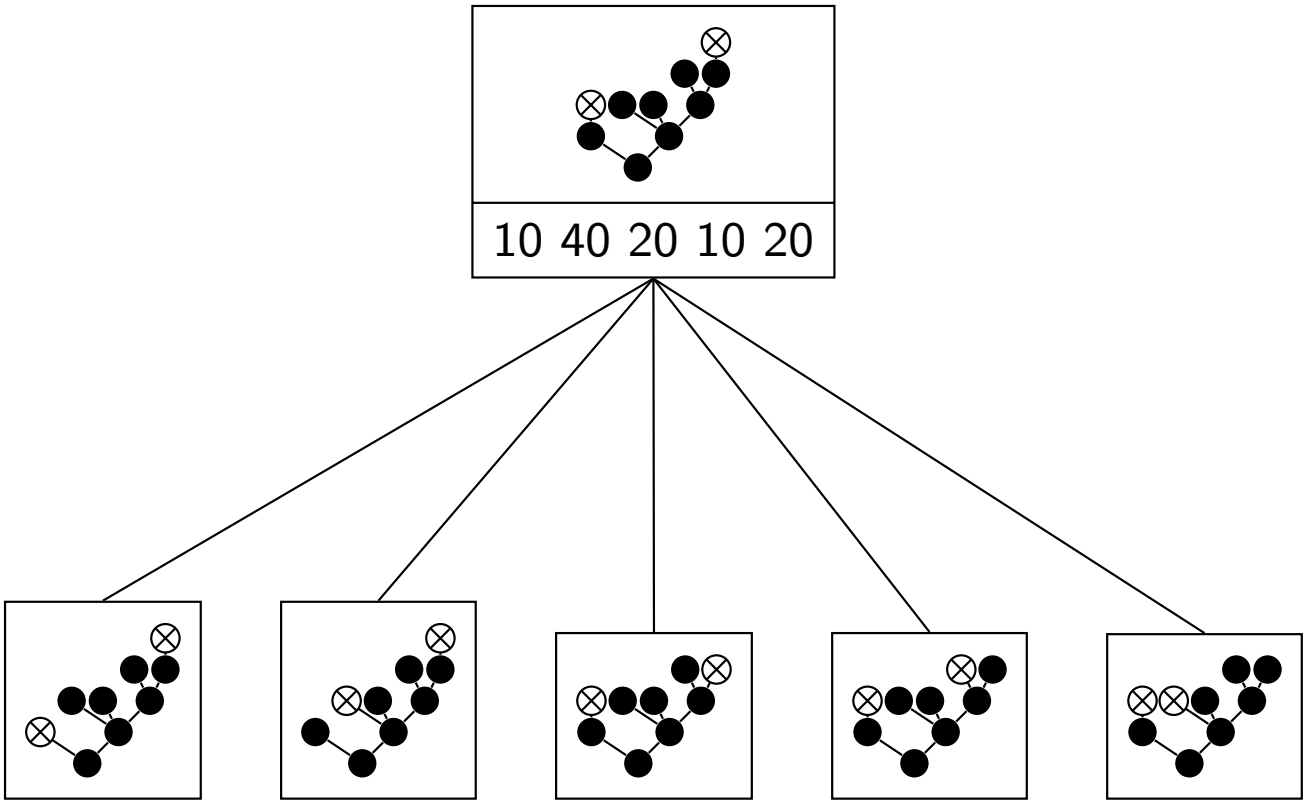
Schedule visualization



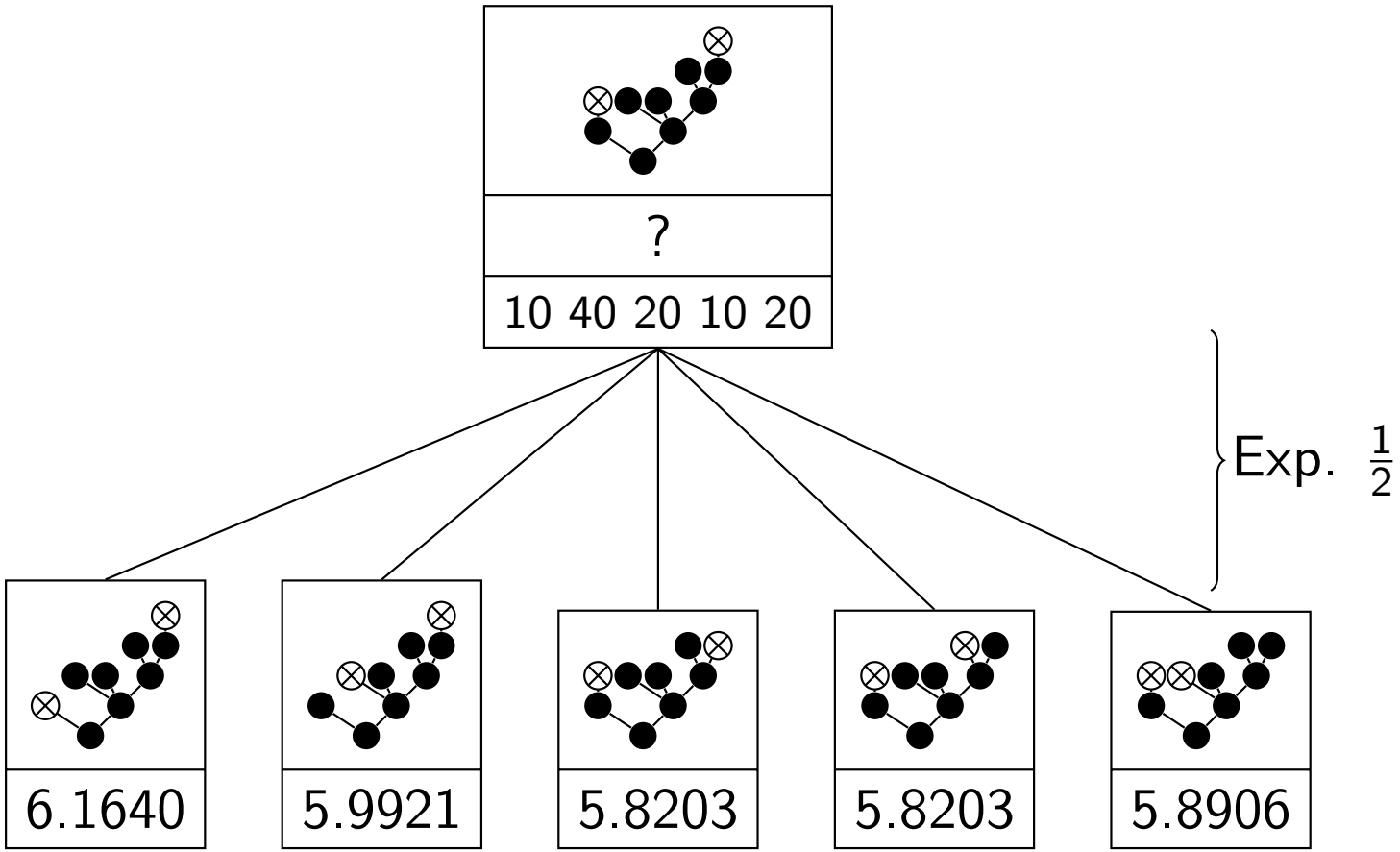
Schedule visualization



Schedule visualization

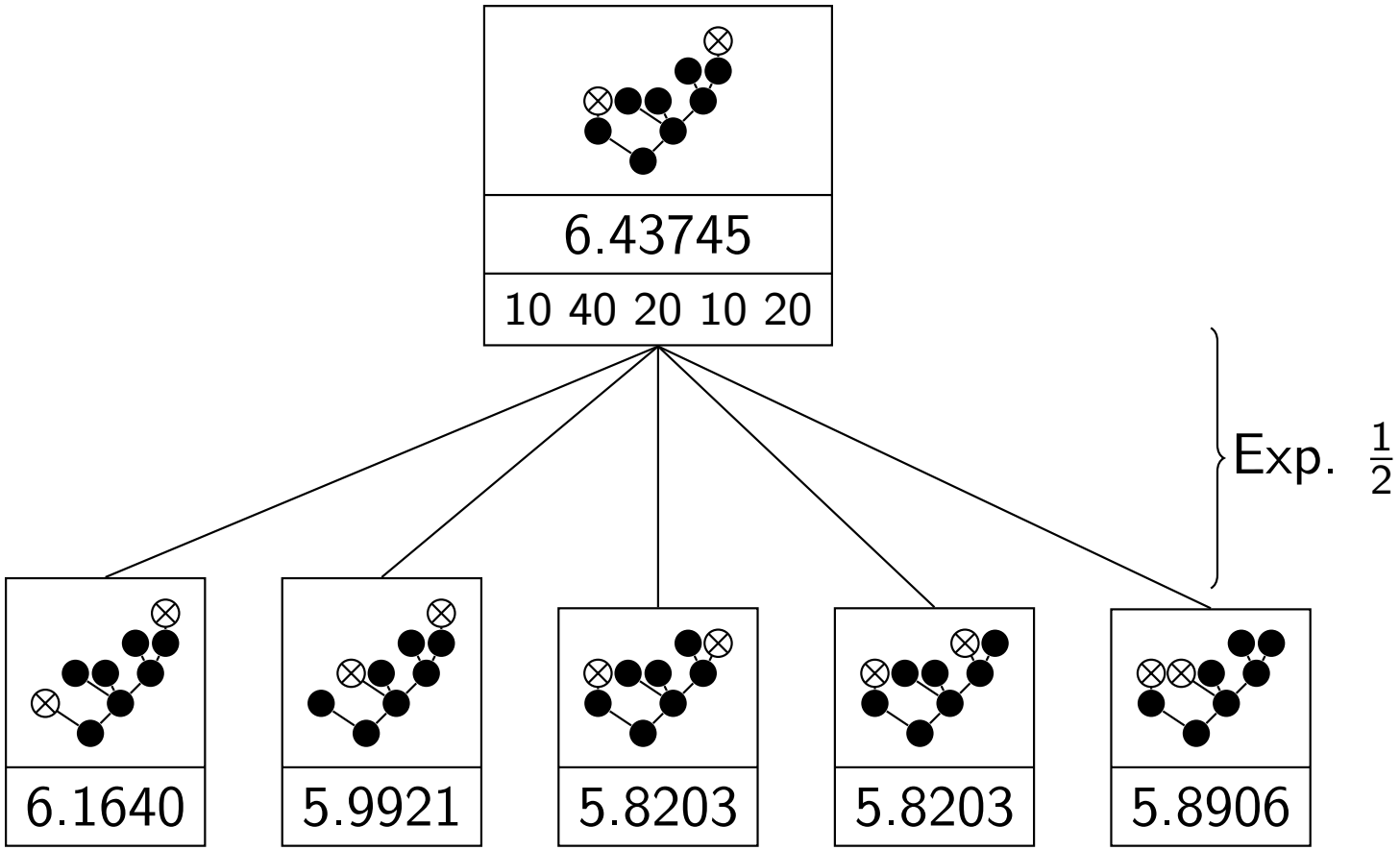


Schedule visualization



$$\frac{1}{2} + 0.1 \cdot 6.1640 + 0.4 \cdot 5.9921 + 0.2 \cdot 5.8203 + \cdots = 6.43745$$

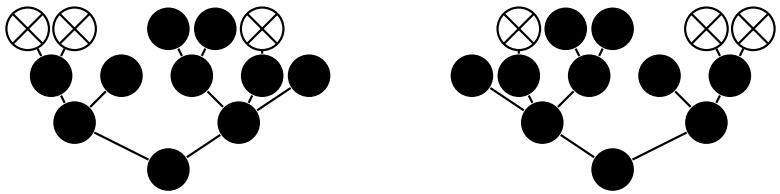
Schedule visualization



$$\frac{1}{2} + 0.1 \cdot 6.1640 + 0.4 \cdot 5.9921 + 0.2 \cdot 5.8203 + \cdots = 6.43745$$

Equivalent snapshots

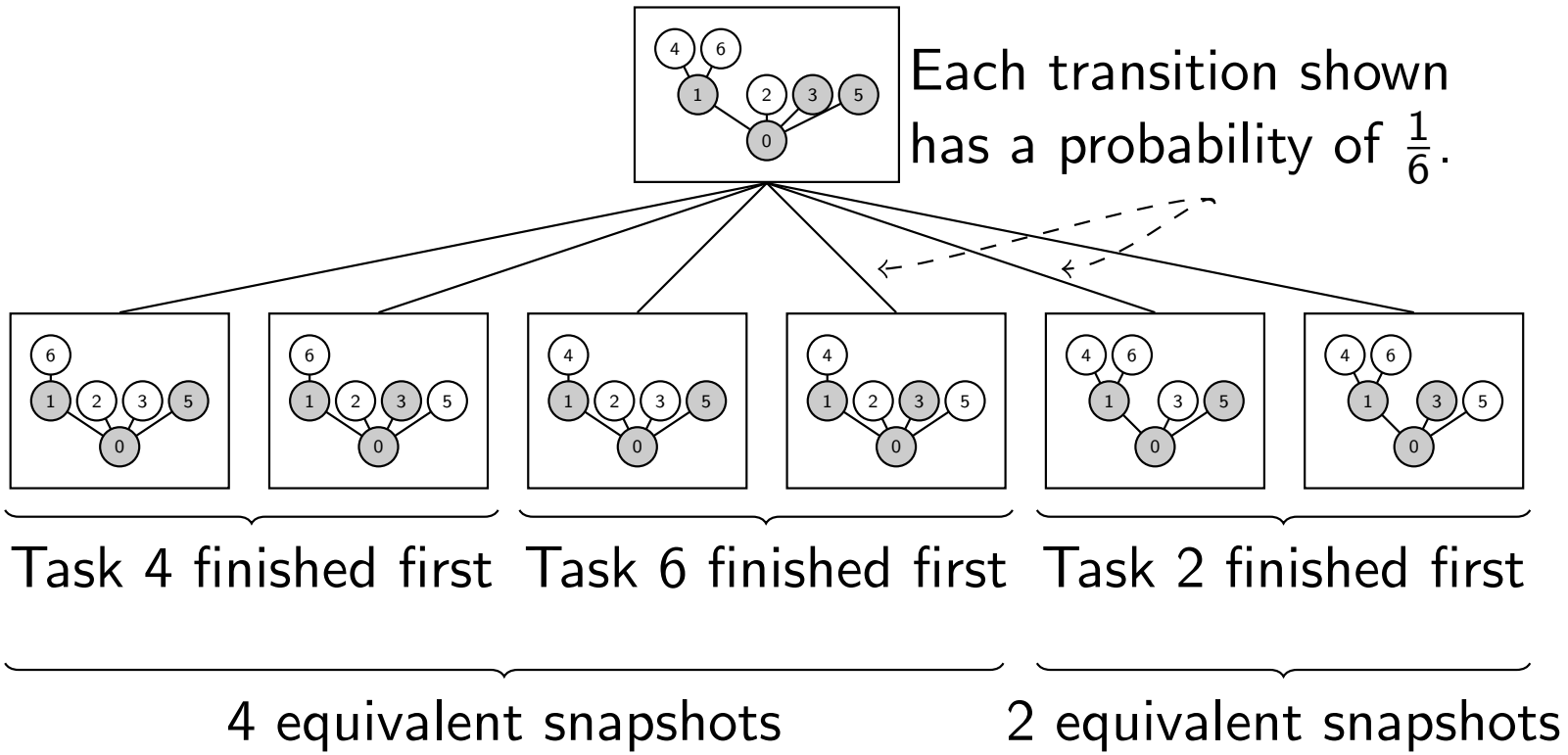
These two snapshots describe the same:



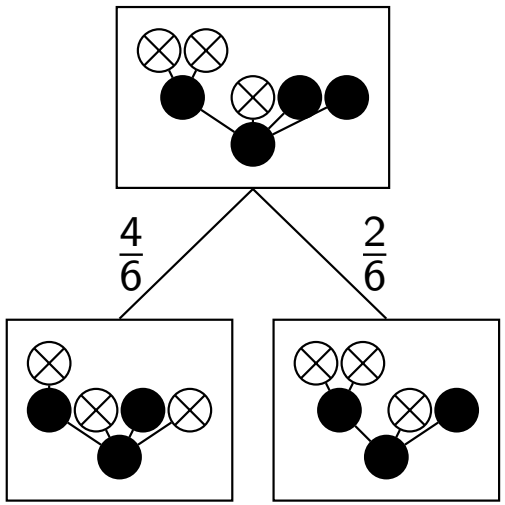
Equivalent snapshots

- Isomorphic intrees
- Scheduled tasks are mapped onto each other
- Equivalent snapshots excluded/re-used if they result from the same finishing task
- *Canonical* snapshots constructable in $O(n)$

Equivalent snapshots — example



Equivalent snapshots — example

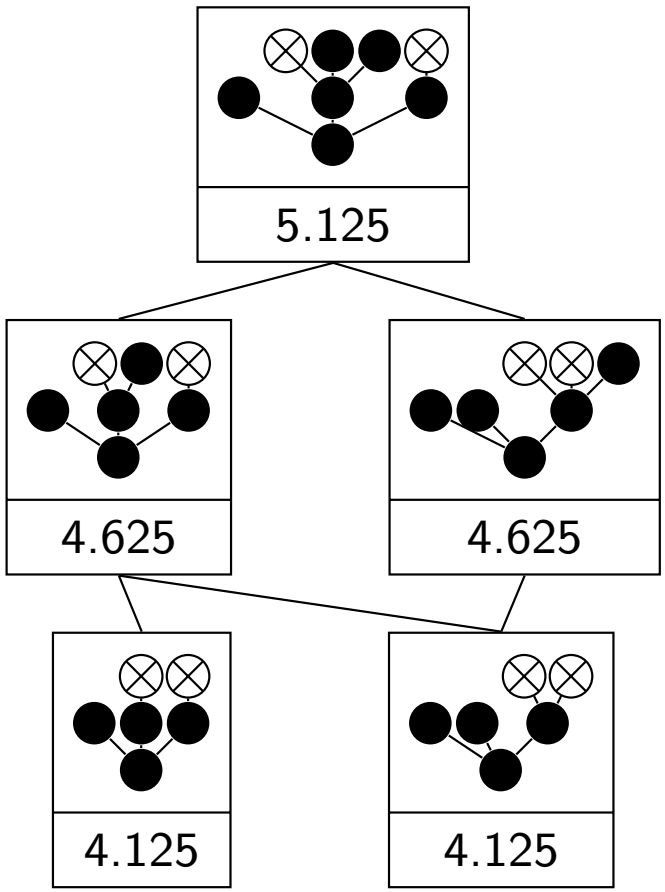


$\Rightarrow$ Reduces number of snapshots

Two processors — optimal solution

Known results (Chandy, Reynolds 1975/1979)

- No deliberate idleness (also for more processors)
- Highest-level-first (HLF) optimal for two processors
- Profile (number of tasks per level) completely determines run time
- Reduce computation of expected run time to profiles
  $\Rightarrow$ Formula computing expected run time for intrees with 2 leaves (Maaß 2001)



Expected run time — special intrees

Theorem

Intrees with profile $\left[\!\left[n_1, (1)^{j-2}, n_j, (1)^{r-j}\right]\!\right]$ *(for* $j \geq 2$*) has expected run time*

$$\mathbb{E}\left[\left[\!\left[n_1, (1)^{j-2}, n_j, (1)^{r-j}\right]\!\right]\right] = r + \frac{A_0(n_1 - 2)}{2^{n_1 - 1}} + \frac{A_{j-1}(n_j - 2)}{2^{n_j + j - 2}},$$

where A_i *is inductively defined as follows:*

$$A_0(n) = (n+1) \cdot 2^n \quad A_{i+1}(n) = \sum_{k=0}^{n} A_i(k)$$

Profile DAG

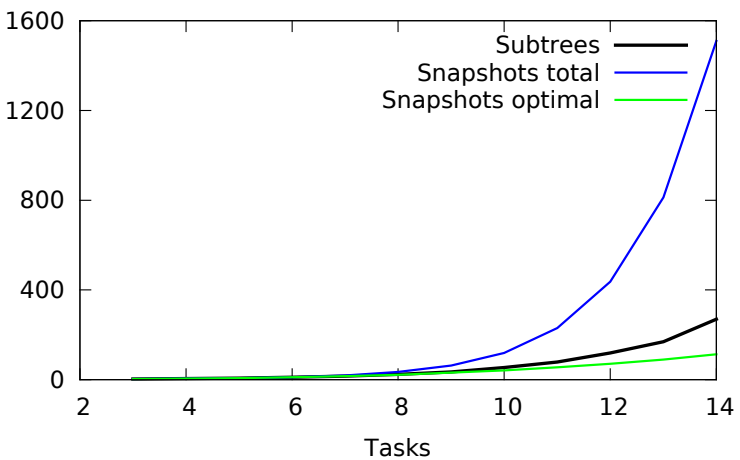
- Use profiles instead of whole snapshots
- Scheduled tasks given implicitly (HLF)
- Worst case profile $\left[\!\left[(1)^{\lfloor \frac{n}{2} \rfloor - 1}, \lceil \frac{n}{2} \rceil, 1\right]\!\right]$
- Worst case profile DAG size has $\lfloor \frac{n}{2} \rfloor \cdot \lceil \frac{n}{2} \rceil + 1$ "profile snapshots"
  $\Rightarrow$ Optimal expected run time computable in polynomial time

Computing optimal schedules

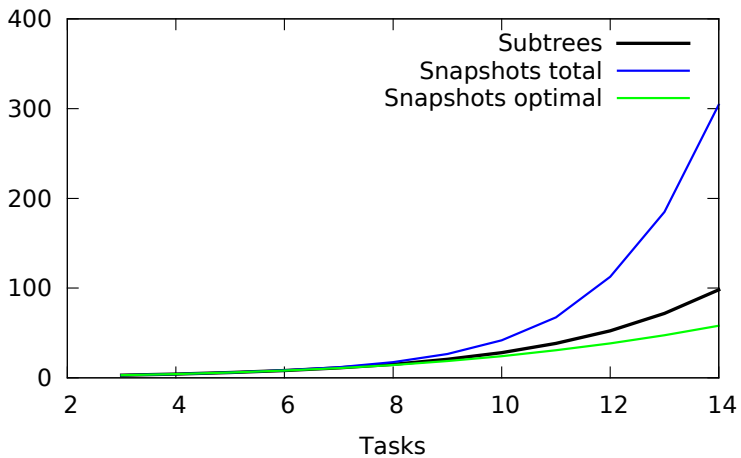
- Exhaustive search (*LEAF* scheduler)
- "Optimize" result from exhaustive search by recursively discarding bad choices
- Exponential run time

Size of the snapshot DAG

- Number of subtrees grows exponentially
  $\Rightarrow$ Number of overall snapshots as well
- Number of snapshots in optimal DAG remarkably smaller



Maximum



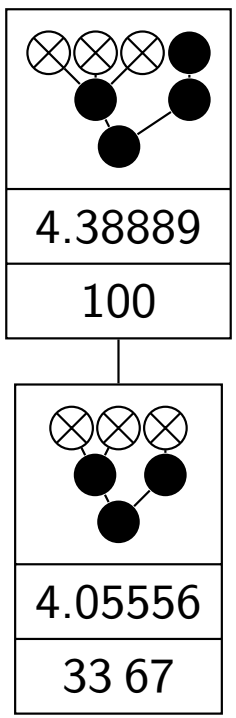
Average

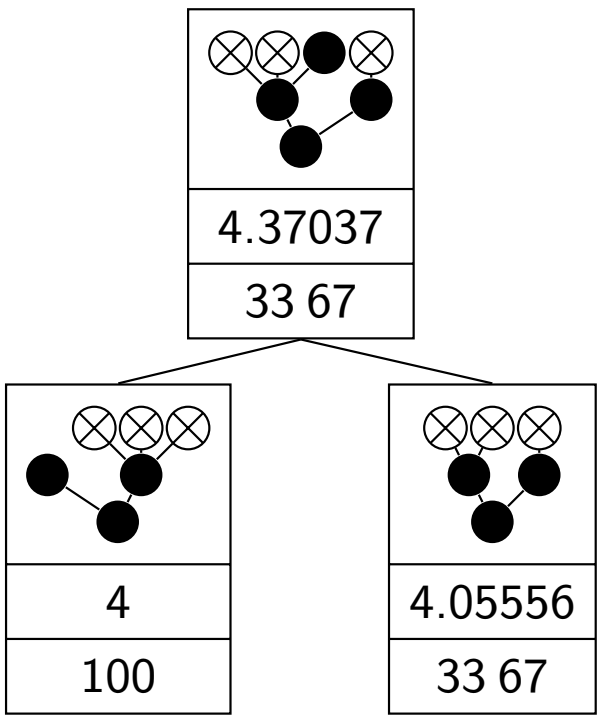
Compressing the snapshot DAG

- Optimal snapshot DAG allows (recursive) merging of some snapshots
- LEAF snapshot DAG prevents us from merging because the run times are not the same for some snapshots

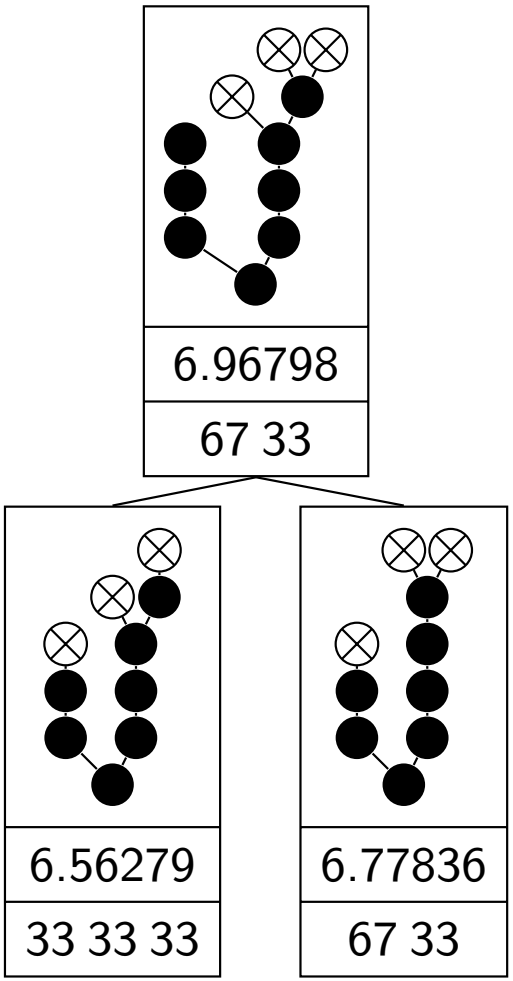
HLF may be ambiguous . . .

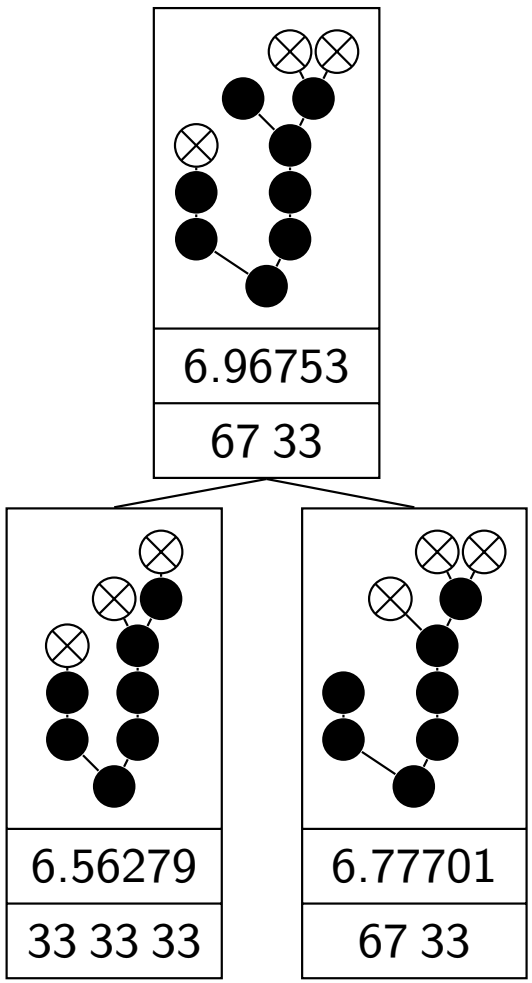
HLF may result in different run times:





. . . or even strictly suboptimal





HLF and (dynamic) list scheduling

- HLF is (like every dynamic list scheduling strategy) suboptimal . . .
- . . . but asymptotically good (Papadimitriou, Tsitsiklis 1987)
- Optimal schedule has to consider previous choices
- Optimal schedule may be one particular run of HLF $\Rightarrow$ HLF is "can-optimal" for these intrees

"2-HLF plus 1"

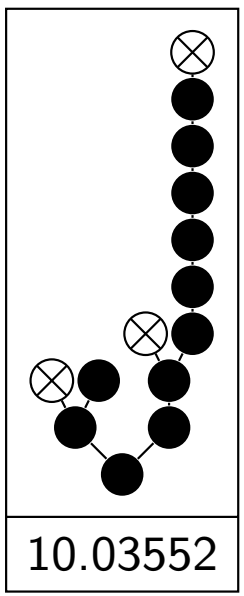
Motivation

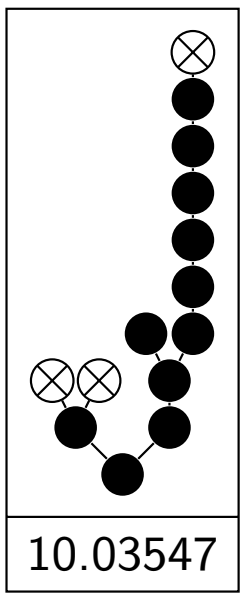
Non-HLF intrees up to 13 tasks have at most one non-HLF task scheduled.

Strategy

Discard snapshots with more than one non-HLF task scheduled.

Counterexample





Only highest and lowest leaves

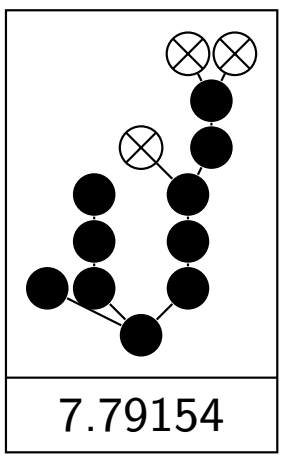
Motivation

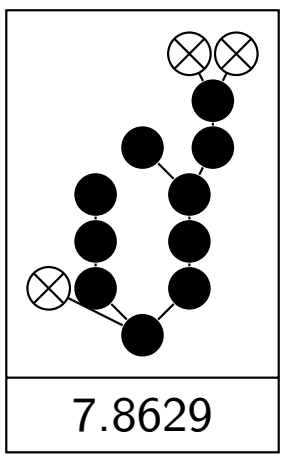
Optimal schedules seen so far scheduled only topmost and lowest leaves.

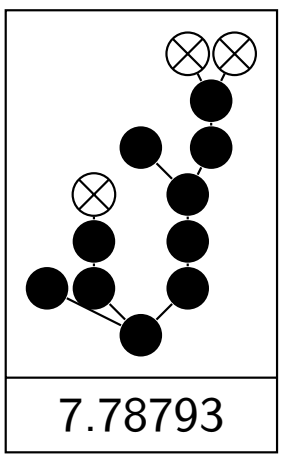
Strategy

Restrict to snapshots where only topmost and lowest leaves are scheduled.

Counterexample







Subtree with fewer topmost tasks

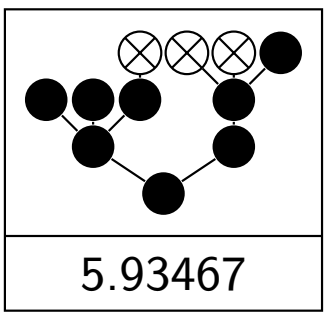
Motivation

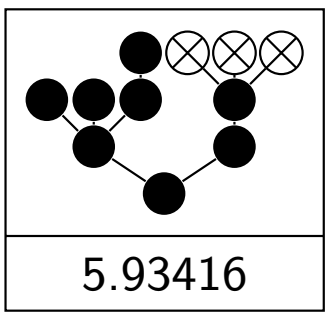
In many cases, a topmost task being the single requirement for its direct successor has to be scheduled

Strategy

Prefer tasks whose corresponding subtree has fewer topmost tasks.

Counterexample



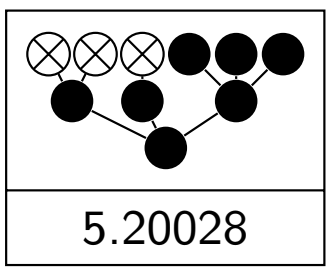


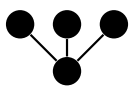
Recursive approach

Strategy

Prefer root's predecessors with maximal processing time.

Counterexample





2.83 resp. 3

2.5

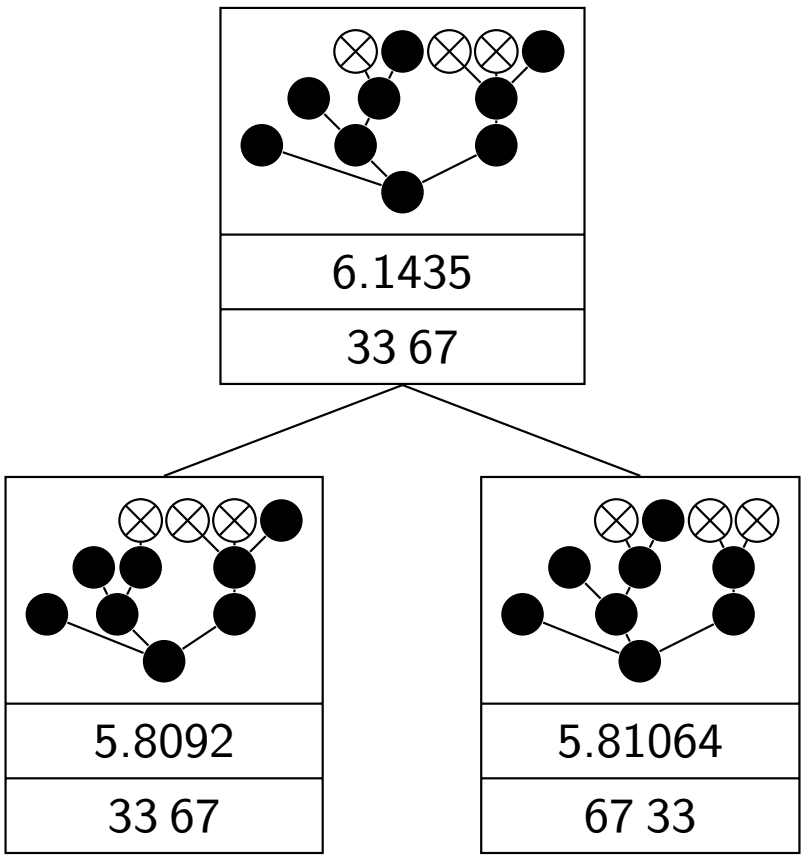
2

Filling up subtrees

Motivation

Subtrees seem to be filled up one after another.

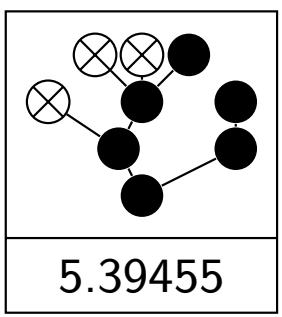
Counterexample



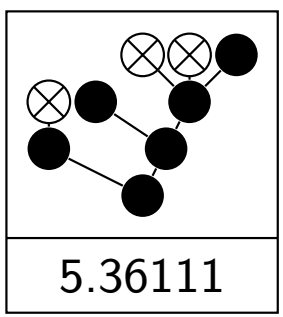
Maximizing 3-processor time, minimizing 1-processor time

Optimal schedules do *not* necessarily . . .

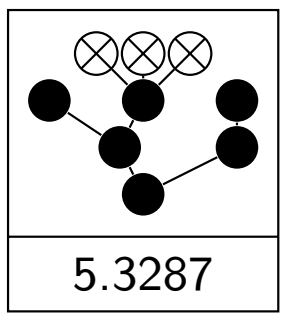
- . . . maximize the expected time span T_3 where 3 processors are busy
- . . . minimize the expected time span T_1 where only 1 processor can work



$T_1 = 2.98$



$T_3 = 2.36$



$T_3 = 2.33, T_1 = 2.99$

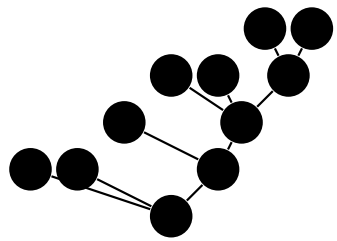
Optimal strategies — conjectures

- *In the beginning*, as many topmost task as possible shall be scheduled
- If the scheduler has a choice, it shall pick a topmost task, if available

Special classes of intrees

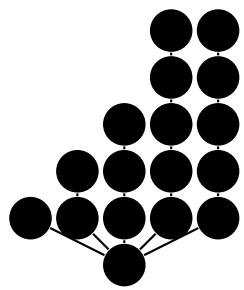
Degenerate intrees

- On each level, at most one task has predecessors
- Degenerate intrees are optimally scheduled by HLF (proven)



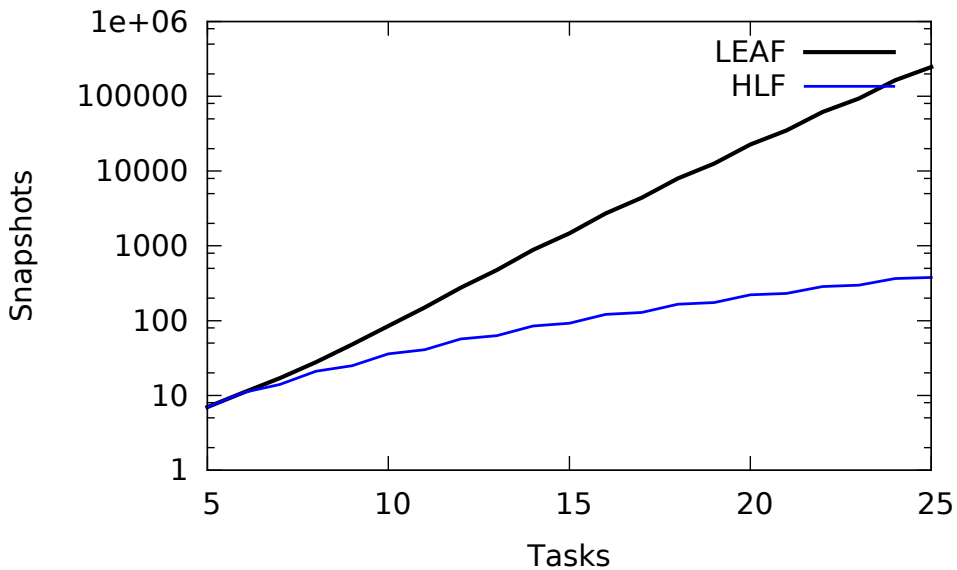
Parallel chains

- Each task, except the root, has at most one predecessor
- Parallel chains with up to 27 tasks are optimally scheduled by HLF



Degenerate intrees and parallel chains — outcome

- HLF is deterministic for these classes . . .
- . . . and needs remarkably less snapshots than LEAF
- Example: Number of snapshots for degenerate binary trees

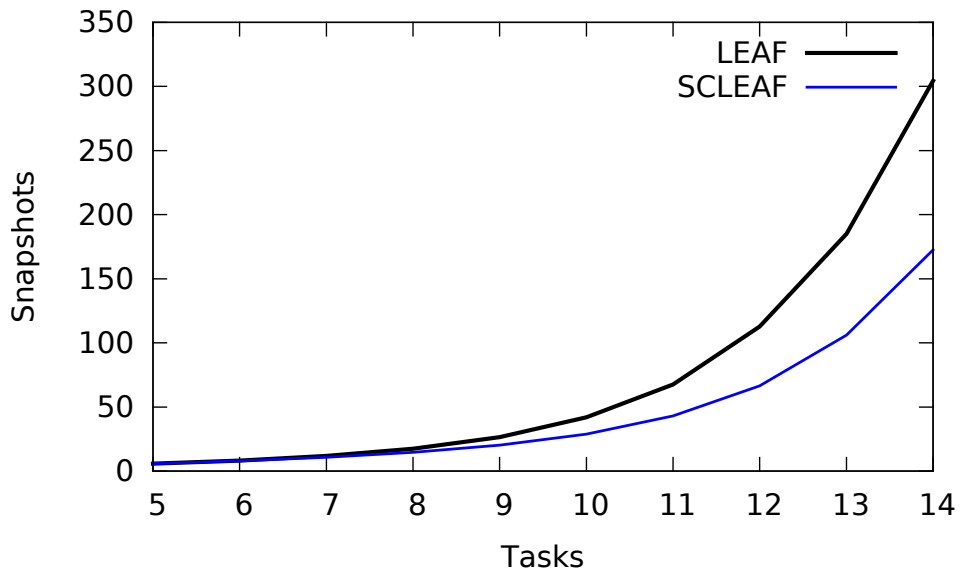


- Analogous results for parallel chains

Improving LEAF with conjectures

SCLEAF

- Simple LEAF scheduler, but . . .
- . . . using HLF when encountering degenerate intrees or parallel chains . . .
- . . . restricts to snapshots where as many topmost task as possible are scheduled



Results

Practical results

- Optimal schedules for all intrees with up to 15 tasks: 11 minutes, less than 2Gb
- More tasks $\Rightarrow$ too much memory $\Rightarrow$ Subdivide into smaller sets of trees
- Optimal schedules for non-trivial intrees with 19 tasks: 1 day

Theoretical results

- New formula for run time for 2 processors
- Conjectures about optimal schedules for 3 processors ...
- ... that could help improving exhaustive search

Features

```
./build/tasks_cs1 -p3 -s leaf --optimize --random 6
--tikz

Normalized: [1, 0] [4, 2, 0] [5, 2, 0] [6, 3, 0]
  !   PTM! [3, 4, 5]  4.27161  (1.90123|1.92593|2.44444)  (
  !    TM! [3, 4, 6]  4.26235  (1.93827|1.86111|2.46296)  (
*H         [4, 5, 6]  4.2037   (2|1.7963|2.40741)         (
Total expected run time: 4.25
Total number of snaps:   16
Writing tikz to default.tex
Compiling to PDF.
Pool sizes: 17, 17,
```

Extensibility

- Different number representations
- Other distributions
- New checks
- Preemptiveness
- Other dependency structures
- Exporters

Further reading: Original problem I

Bruno, John (1985). "On scheduling tasks with exponential service times and in-tree precedence constraints". English. In: *Acta Informatica* 22.2, pp. 139–148. ISSN: 0001-5903.

Chandy, K. M. and P. F. Reynolds (1975a). "Scheduling partially ordered tasks with probabilistic execution times". In: *Proceedings of the fifth ACM symposium on Operating systems principles*. SOSP '75. New York, NY, USA: ACM, pp. 169–177.

— (Nov. 1975b). "Scheduling partially ordered tasks with probabilistic execution times". In: *SIGOPS Oper. Syst. Rev.* 9.5, pp. 169–177. ISSN: 0163-5980.

— (1979). *Scheduling Partially Ordered Tasks with Exponentially Distributed Times*.

Further reading: Original problem II

Maaß, Moritz G. (Sept. 2001). "Scheduling Independent and Identically Distributed Tasks with In-Tree Constraints on Three Machines in Parallel". MA thesis. Lehrstuhl für Effiziente Algorithmen, Institut für Informatik, TU München.
Pinedo, M. and G. Weiss (1985). "Scheduling Jobs with Exponentially Distributed Processing Times and Intree Precedence Constraints on Two Parallel Machines". In: *Operations Research* 33 (6), pp. 1381–1388.

Further reading: Equivalent snapshots

Aho, A.V., J.E. Hopcroft, and J.D. Ullman (1974). *The design and analysis of computer algorithms*. Addison-Wesley series in computer science and information processing. Addison-Wesley Pub. Co.

Matula, D.W. (1968). "A natural rooted tree enumeration by prime factorization". In: *SIAM Review* 10, p. 273.

Zemlyachenko, V.N., N.M. Korneenko, and R.I. Tyshkevich (1985). "Graph isomorphism problem". English. In: *Journal of Soviet Mathematics* 29.4, pp. 1426–1481. ISSN: 0090-4104.

Further reading: Scheduling I

Coffman E.G., Jr. and R.L. Graham (1972). "Optimal scheduling for two-processor systems". English. In: *Acta Informatica* 1.3, pp. 200–213. ISSN: 0001-5903.

Garey, M. R., D. S. Johnson, and Ravi Sethi (1976). "The Complexity of Flowshop and Jobshop Scheduling". In: *Mathematics of Operations Research* 1.2, pp. 117–129.

Graham, R.L. et al. (1979). "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey". In: *Discrete Optimization II Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium co-sponsored by IBM Canada and SIAM Banff, Aha. and Vancouver*. Ed. by E.L. Johnson P.L. Hammer and B.H. Korte. Vol. 5. Annals of Discrete Mathematics. Elsevier, pp. 287 –326.

Further reading: Scheduling II

Pinedo, M. (2008). *Scheduling: Theory, Algorithms, and Systems*. Springer. ISBN: 9780387789347.