One Step towards Bridging the Gap between Theory and Practice in Moldable Task Scheduling with Precedence Constraints

Sascha Hunold

Research Group Parallel Computing
Institute of Information Systems
Vienna University of Technology
Austria

July 2, 2014

Motivation - Divergence of Scheduling Research

- parallel machine scheduling
  - complex architecture of parallel machines / software (OS)
  - (almost) impossible to know much about the problem
    - job sizes (execution times)
    - release dates, etc.

Motivation - Divergence of Scheduling Research

- parallel machine scheduling
  - complex architecture of parallel machines / software (OS)
  - (almost) impossible to know much about the problem
    - job sizes (execution times)
    - release dates, etc.
- theoreticians
  - "an interesting problem, we need novel insights"
    - let us use a simplistic model
  - "I found an FPTAS for the simplistic model. I have a complicated 100-page proof. Problem solved."

Motivation - Divergence of Scheduling Research

- parallel machine scheduling
  - complex architecture of parallel machines / software (OS)
  - (almost) impossible to know much about the problem
    - job sizes (execution times)
    - release dates, etc.
- theoreticians
  - "an interesting problem, we need novel insights"
    - let us use a simplistic model
  - "I found an FPTAS for the simplistic model. I have a complicated 100-page proof. Problem solved."
- practitioners
  - "I forgot my Turing machine today"
  - "How do I adapt your FPTAS to 3 levels of cache on 200,000 NUMA cores interconnected via 6D torus network under a certain workload and background system noise?"
  - "I also figure that your $O(n^{60})$ DP could be a little slow."
  - "But actually, I do not care about theoretical results. I will simply reinvent the wheel and sell it as breakthrough."

Situation

Theoretical Results .... VOID .... Practical Results

The Problem

- notation follows "Scheduling for Parallel Processing" by Drozdowski [Dro09]
- types of parallel tasks
  - rigid
  - moldable
  - malleable
- precedence constraints between n tasks
  - direct acyclic graph (DAG)
- schedule n moldable tasks on m identical processors
- in Graham's 3-field notation
  - $P|any, NdSub, prec|C_{\max}$ and $P|any, prec|C_{\max}$
  - any - moldable tasks
  - $NdSub$ - nondecreasing sublinear speedup
  - $prec$ - precedence constraints

Common Assumptions - NdSub ?

- PROC-TIME-NON-INCREASING
  The processing time $p(l)$ of a moldable task J is non-increasing in the number l of the processors allotted to it, that is, $p(l) \leq p(l')$, for $l \geq l'$;

Common Assumptions - NdSub ?

- PROC-TIME-NON-INCREASING
  The processing time $p(l)$ of a moldable task J is non-increasing in the number l of the processors allotted to it, that is, $p(l) \leq p(l')$, for $l \geq l'$;
- WORK-NON-DECREASING
  The work W(l) = w(p(l)) = lp(l) of a moldable task J is non-decreasing in the number l of the processors allotted to it, that is, $W(l) \leq W(l)$ for $l \leq l'$.

Common Assumptions - NdSub ?

- PROC-TIME-NON-INCREASING
  The processing time $p(l)$ of a moldable task J is non-increasing in the number l of the processors allotted to it, that is, $p(l) \leq p(l')$, for $l \geq l'$;
- WORK-NON-DECREASING
  The work W(l) = w(p(l)) = lp(l) of a moldable task J is non-decreasing in the number l of the processors allotted to it, that is, $W(l) \leq W(l)$ for $l \leq l'$.
- PROC-TIME-STRICTLY-DECREASING
  The processing time $p(l)$ of a moldable task is strictly decreasing in the number l of the allocated processors: $p(l) < p(l')$, for $l > l'$.

Common Assumptions - NdSub ? (cont'd)

- SPEEDUP-CONCAVE "The first restriction is that all speedup functions are concave at least between 0 and the processor number $\hat{p}$ where the maximal speedup is reached." [SS12]

Common Assumptions - NdSub ? (cont'd)

- SPEEDUP-CONCAVE "The first restriction is that all speedup functions are concave at least between 0 and the processor number $\hat{p}$ where the maximal speedup is reached." [SS12]
- WORK-CONVEX-PROC-TIME
  The work function $w(p(l))$ is convex in the processing time $p(l)$.

Related Work - Lepère, Trystram, Woeginger [LTW02]

- assumptions
  1. PROC-TIME-NON-INCREASING
  2. WORK-NON-DECREASING
- decouple scheduling problem
- allotment problem (MT-ALLOTMENT)
  - Skutella's linear relaxation of discrete time-cost trade-off problem (DTCT)
  - 2 approximation
- mapping problem (MT-MAKESPAN)
  - Graham's list scheduling for parallel tasks
    - earliest possible task first
  - the proof is the trick here
  - construct a heavy path in the transitive closure of the DAG
- approximation ratio: $3 + \sqrt{5} \approx 5.23606$

Related Work - Jansen and Zhang [JZ06]

- same assumptions as [LTW02]
  1. PROC-TIME-NON-INCREASING
  2. WORK-NON-DECREASING
- provide linear program formulation
- kept mapping function
- the trick here is to find the right rounding parameter ρ
  - round fractional solution to feasible allotment
- approximation ratio: ≈ 4.730598

Related Work - Jansen and Zhang [JZ12]

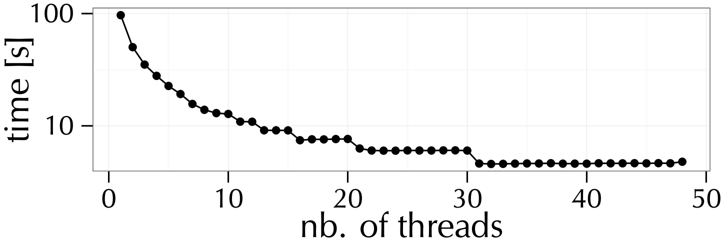
- assumptions:
  1. PROC-TIME-NON-INCREASING
  2. WORK-NON-DECREASING
  3. SPEEDUP-CONCAVE
- allotment: reformulate the LP
  - essence: use a variable indicating that a job is (fractionally) allocated to l processors
  - constraint: $\sum_{l=1}^{m} x_{j,l} = 1$
  - interesting here: at most two $x_{j,l}$ are non-zero and adjacent
- mapping step unchanged
- approximation ratio: ≈ 3.291919

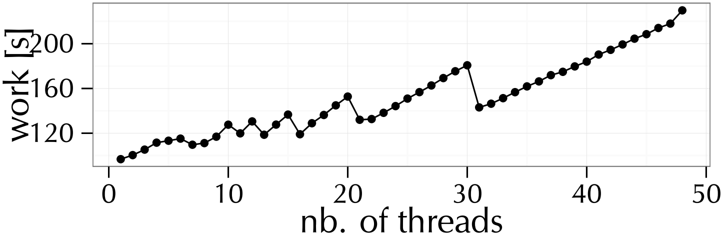
Related Work - Chen and Chu [CC13]

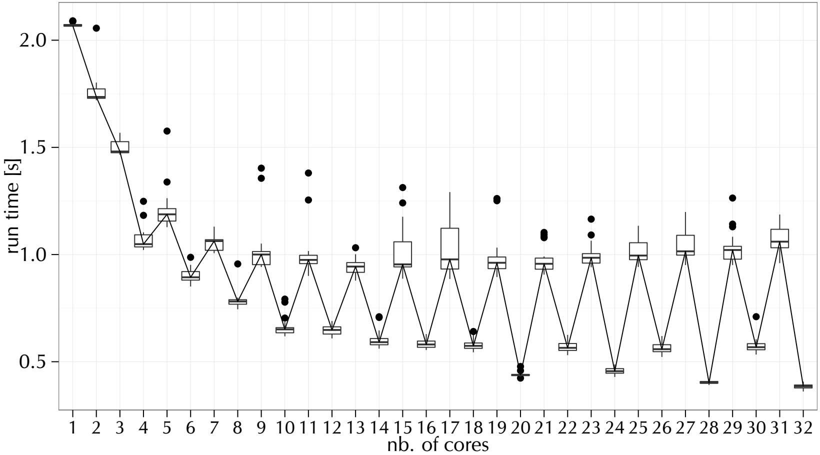
- an algorithm (heavily) based on JZ06
- assumptions
  1. PROC-TIME-NON-INCREASING
  2. WORK-NON-DECREASING
  3. WORK-CONVEX-PROC-TIME
- allotment:
  - precompute the work of each possible allocation for all tasks
  - use this information to add constraints to LP (which later help in the rounding step)
- mapping unchanged
- approximation ratio: ≈ 3.4142
- then, adding
  1. PROC-TIME-STRICTLY-DECREASING
  2. leads to approximation ratio: 2.9549

Reality 1

NAS PB - LU benchmark (4 sockets, 48 cores, AMD Opteron 6168)







MPICH 3.0.4, 32x1 (process per node)

CPA13 heuristic

- based on CPA by Radulescu and van Gemund [RvG01]
- main purpose: schedule tasks with arbitrary speed-up functions
- allotment solution, ingredients:
  - consider only allocations which provide a relative runtime gain of $x\,\%$
  - force task parallel execution of tasks
  - iteratively add processors to tasks on critical path
  - until $L_{CP} < W/m$
- mapping solution
  - similar to list scheduling approach of Lepère, Trystram, Woeginger
  - but prioritize tasks by bottom level (length of path to sink)
  - but consider packing of tasks if estimated completion time is not increased (binary search to find possibly smaller allotment)

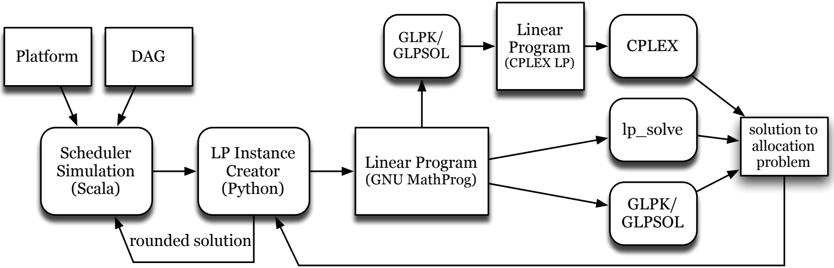
General Questions

- How good are "frequently cited" heuristics (e.g., CPA) compared to approximation algorithms?
- How fast are current LP solvers (e.g., CPLEX) for solving "practically relevant" problems?

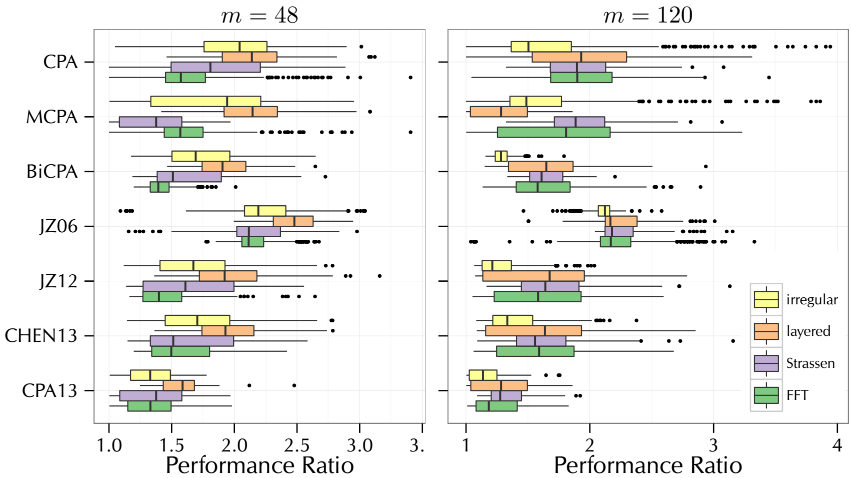
algorithm	allocation	mapping
CPA13	$O(nm(n+e))$	$O(n(\log n + m \log m) + e)$
JZ06	$O(LP(mn, n^2 + mn))$	$O(mn)$
JZ12	$O(LP(mn, n^2 + mn))$	$O(mn)$
Chen13	$O(LP(mn, n^2 + mn) + mn)$	$O(mn)$

- let us fulfill all assumptions
  - PROC-TIME-NON-INCREASING, WORK-NON-DECREASING, PROC-TIME-STRICTLY-DECREASING, SPEEDUP-CONCAVE, WORK-CONVEX-PROC-TIME

Experimental Setup

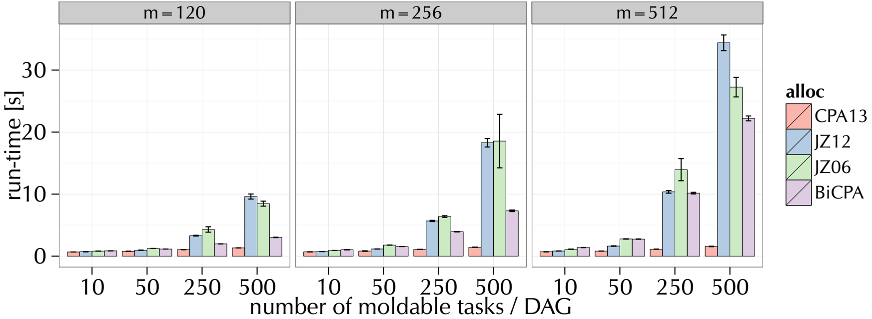


Simulation Results - Distribution of Makespans



performance ratio $\hat{=}\ C_{\text{max}}/LB$

Scalability



- Intel i7-3770 @ 3.40 GHz, 4 cores / 8 hardware threads
- CPLEX Studio 12.5.1 Linux x86-64
  - LP programs use all 4 cores

But can I be sure that the results are correct?

Missing Constraint - [JZ06]

$$
\begin{array}{lll}
\text{min} & C & \\
\text{such that} & 0 \le C_j \le L & \text{for all } j \\
& C_j + x_k \le C_k & \text{for all } j \text{ and } k \in \Gamma^+(j) \\
& x_j \le C_j & \text{for all } j \\
& x_j \le p_j(1) & \text{for all } j \\
& x_{j_i} \le x_j & \text{for all } j \text{ and } i = 1, \dots, m \\
& 0 \le x_{j_i} \le p_j(i) & \text{for all } j \text{ and } i = 1, \dots, m-1 \\
& x_{j_m} = p_j(m) & \text{for all } j \\
& \hat{w}_j(x_j) = \sum_{i=1}^{m} \bar{w}_{j_i}(x_{j_i}) & \text{for all } j \\
& P = \sum_{j=1}^{n} p_j(1) & \\
& \sum_{j=1}^{n} \hat{w}_j(x_j) + P \le W & \\
& L \le C & \\
& W/m \le C & \\
& \bar{w}_j(x_{j_m}) = 0 & \text{for all } j \\
& \bar{w}_{j_i}(x_{j_i}) = [W_j(i+1) - W_j(i)] \dfrac{p_j(i) - x_{j_i}}{p_j(i)} & \text{for all } j \text{ and } i = 1, \dots, m-1
\end{array}
$$

Missing Constraint (2) - [CC13]

- LP of Chen and Chu also misses same constraint
- for assumption PROC-TIME-STRICTLY-DECREASING paper provides another rounding procedure leading to smaller bound
  - problem: rounding procedure described in paper for strictly decreasing function produced very large $C_{\max}$'es (why?)
- run-time of this LP was huge
  - reason: LP uses a possibly different set of processor allocations for each moldable task (many more constraints)

Summary of Problems Occurred

- missing constraints in linear programs
  - very time-consuming to detect (at least for me)
- the problem of precision: floats/double
  - "64 Bit is finite"
  - problem generator
    - choose the execution time of tasks (for 1 proc) randomly and apply some strong strong scaling function
    - $0.00000000000001345 \not< 0.00000000000001345$

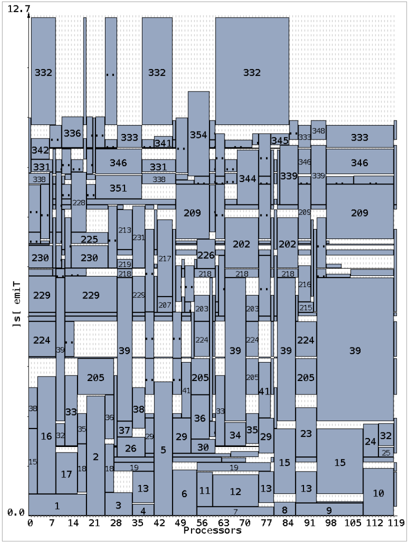
Summary of Problems Occurred

- missing constraints in linear programs
  - very time-consuming to detect (at least for me)
- the problem of precision: floats/double
  - "64 Bit is finite"
  - problem generator
    - choose the execution time of tasks (for 1 proc) randomly and apply some strong strong scaling function
    - $0.00000000000001345 \not< 0.00000000000001345$
- many sources of errors
  - me
  - DAG generator
  - platform generator
  - predictor for execution time of moldable tasks
  - translation of linear program in mathematical notation to AMPL/MathProg
  - parsing results from LP solver
  - implementation of mapping algorithm / simulator

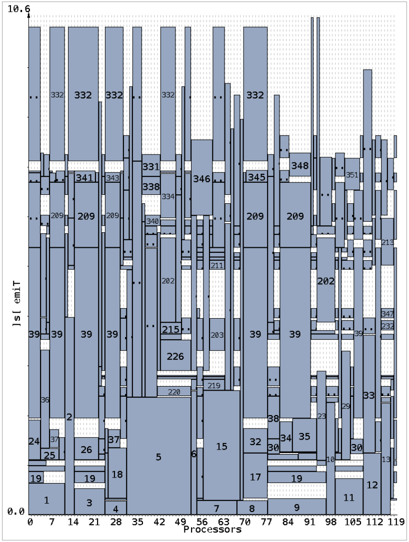
Summary of Problems Occurred

- missing constraints in linear programs
  - very time-consuming to detect (at least for me)
- the problem of precision: floats/double
  - "64 Bit is finite"
  - problem generator
    - choose the execution time of tasks (for 1 proc) randomly and apply some strong strong scaling function
    - $0.00000000000001345 \not< 0.00000000000001345$
- many sources of errors
  - me
  - DAG generator
  - platform generator
  - predictor for execution time of moldable tasks
  - translation of linear program in mathematical notation to AMPL/MathProg
  - parsing results from LP solver
  - implementation of mapping algorithm / simulator
- consequence: debugging is a nightmare

Linear Program Solvers – Primal vs. Dual @ JZ06



`--primal`



`--dual`

Summary - What Did I Miss in this Study?

- help from theoreticians
- public database with codes (e.g., LPs)
- set of benchmarks
  - relevant problems (for DAGs of moldable tasks)
  - possibly optimal solutions for small instances
  - implementations of algorithms (give me the sources)
  - some example schedules produced by algorithms
- wish list (from the view of a practitioner)
  - Complexity results for scheduling problems
    - `http://www.informatik.uni-osnabrueck.de/knust/class/`
  - add implementation and test cases
- my source code is available upon request

References I

[CC13] Chi-Yeh Chen and Chih-Ping Chu.
A 3.42-approximation algorithm for scheduling malleable tasks under precedence constraints.
IEEE Transactions on Parallel Distributed Systems, 24(8):1479–1488, 2013.

[Dro09] Maciej Drozdowski.
Scheduling for Parallel Processing.
Springer, 2009.

[JZ06] K Jansen and H Zhang.
An Approximation Algorithm for Scheduling Malleable Tasks under General Precedence Constraints.
ACM Transactions on Algorithms, 2(3):416–434, 2006.

[JZ12] Klaus Jansen and Hu Zhang.
Scheduling malleable tasks with precedence constraints.
Journal of Computer and System Sciences, 78(1):245–259, 2012.

[LTW02] R. Lepère, D. Trystram, and G.J. Woeginger.
Approximation Algorithms For Scheduling Malleable Tasks Under Predence Constraints.
International Journal of Foundations of Computer Science, 13(04):613–627, 2002.

References II

[RvG01] A. Radulescu and A.J.C. van Gemund.
A Low-Cost Approach towards Mixed Task and Data Parallel Scheduling.
In *ICPP*, pages 69–76, 2001.

[SS12] Peter Sanders and Jochen Speck.
Energy Efficient Frequency Scaling and Scheduling for Malleable Tasks.
In *Proc. of the Euro-Par*, pages 167–178, 2012.