Parameterised jobshop scheduling problems

Peter Habermehl (IRIF, Paris)
Ongoing work with A. Sangnier (IRIF) and G. Zetzsche (MPI-SWS, Kaiserslautern)

SYNCOP 2019
Jobshop scheduling problems

• Well known combinatorial optimisation problems
• (finite number of) jobs
• (finite number of) machines
• Each job has to accomplish some task
• which consists of operations which use some machine(s)
• A machine can only be used by one job at the same time
• The operations must obey ordering constraints
Jobshop scheduling problems

- What is the optimal schedule?
- Easily computable by trying all schedules
- Typically NP-hard
- Here: a parameterised version
Parameterised jobshop scheduling

- A fixed number of machines \({\{a, b, c, \ldots}\}\)
- A parameterised number of identical jobs
- Each job is given as a sequence of the machines it has to use successively
- For example: \(a.a.b.c.d.a.a.b.c.c.d\)
- Each machine can be used by one process at a given moment.
- Each step costs 1
Main problem

- Given a number of machines $n$, compute $\text{cost}(n) := \text{the number of total steps to complete all } n \text{ jobs}$
- Obviously, $\text{count}(n)$ can be computed for fixed $n$
- We want to compute a representation of $\{(n,\text{count}(n)) \mid n \geq 1\}$ in one shot
Example

\[ \text{a a b a b b} \]

\[ \begin{array}{cccc}
  \text{a a b a b b} & \\
  \text{a a b a b b} & \\
  \text{a a b a b b} & \\
  \text{a a b a b b} & \\
  \text{a a b a b b} & \\
  \text{a a b a b b} & \\
\end{array} \]
Example

- a.a.b.a.b.b
- Upper bound for cost(n): 6*n,
  - since each job takes at most 6 time units
- Lower bound for cost(n): 3*n
  - since each job must use a at least 3 times
- Therefore, 3*n <= cost(n) <= 6*n
- Here, cost(n) = 3*n+3
Some special cases

- If the job $j$ uses the same machine all the time:
  \[\text{cost}(n) = |j| \times n\]

- If the job uses $|j|$ different machines:
  \[\text{cost}(n) = n + |j| - 1\]
In general

• Let $j$ be a job
• Let $f$ be the length of $j$
• Let $m$ be one of the machines which is used the most
• Let $g$ be the number of times $m$ is used
• Clearly, $g \times n \leq \text{cost}(n) \leq f \times n$
• we show that $\text{cost}(n) \leq g \times n + c$ for some constant $c$
Main result

- cost(n) is a semilinear function
  - *(n, cost(n)) | n >= 0* is a semilinear set
  - that means:
    \[
    \begin{align*}
    \text{cost}(n) &= \begin{cases} 
        d_1 & \text{if } n = 1 \\
        \ldots & \\
        d_p & \text{if } n = p \\
        k\cdot n + c_1 & \text{if } n \mod q = 0 \\
        k\cdot n + c_2 & \text{if } n \mod q = 1 \\
        \ldots & \\
        k\cdot n + c_q & \text{if } n \mod q = q-1
    \end{cases}
    \end{align*}
    \]

- Solution: transformation to a Petri Net problem
Transformation to a PN problem

- Counting abstraction
  - Each position in the job corresponds to a control state
  - Consider number of jobs in each state
- Construct an equivalent PN N
  - Each position in the job corresponds to a place in N
  - Transitions of N are moving tokens ahead
  - Each transition is labeled by the corresponding set of machines
- Initially, \( n \) tokens or a generating transition
- Each transition is counted for the cost
Example

• a.a.b…. 
Transition invariants

• Let $M$ be the incidence matrix of $N$

• A transition invariant is a vector $t$ (multiplicities of transitions)

  such that $Mt = 0$

• Executing a sequence of transitions corresponding to $t$ keeps the token counts constant
Transition invariants

Example:
Transition invariants

• Here all transition invariants $t$ are realisable
• which means, there exists a reachable marking, s.t. from there a sequence of transitions with count $t$ can be executed
Transition invariants

• Example
Transition invariants

- One can compute all transition invariants
  - finite number of minimal transition invariants
- Compute **optimal** transition invariants
  - the machine $m$ is always in use
- For any number $n$ we can construct a run where almost all the time transitions from an optimal transition invariant are used
Example

- Optimal transition invariant:

- Realisation:
Computing cost(n)

- We obtain $k \times n \leq \text{cost}(n) \leq k \times n + c$
- It remains to compute for each $c'$ with $0 \leq c' \leq c$:
  \[ \{n \mid \text{cost}(n) = k \times n + c'\} \]
- Modify PN $N$:
  - Generate $k \times n + c'$ tokens in a “counting” place and $n$ tokens in the initial place
  - Remove one token of the “counting” place for each transition
  - Define a PN language with one-letter: reach empty marking
  - Since one-letter PN languages are regular (Hauschildt/Jantzen 94), we have that $\{n \mid \text{cost}(n) = k \times n + c'\}$ is semilinear.
Boundedness conjecture

• For each execution of the PN N, there is an execution with same or better cost, where the number of tokens are bounded

• Would imply easily the result
Extensions

- Steps which cost different from 1:
  - Cost $k$: $k$ steps of cost 1
  - Rescheduling

- A job can choose from several sequences:
  - For example: aababb or bbabab or aabb or bbaa
  - Here we still have just one parameter $n$
  - The same reasoning can be applied
Extensions

- Several parameters:
  - $n_1$ jobs of type 1, $n_2$ jobs of type 2, etc.
  - Compute $\text{cost}(n_1,n_2,...,n_i)$
  - We still have that the optimal cost can be computed up to a constant $c$
- but the same reasoning as with $\text{cost}(n)$ can not be applied