Multiprocessor Scheduling Problem with Communication Delays

Tanja Davidović, **Leo Liberti**, Nelson Maculan, Nenad Mladenović, Pedro Teixeira

SANU, Belgrade, Serbia and Montenegro

LIX, École Polytechnique, France

COPPE, Rio de Janeiro, Brazil

Summary of Talk

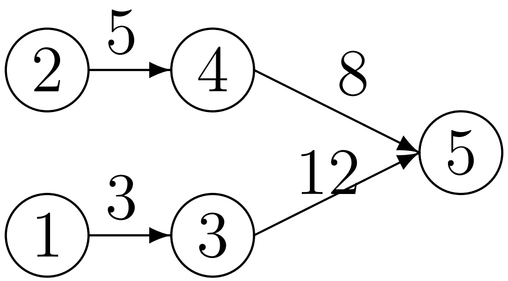
- Problem definition
- Classic formulation
- Usual vs. compact linearization
- Packing formulation
- Computational results

Problem definition

- Scheduling problem: assign n tasks to an architecture of p homogeneous processors such that the makespan is minimized
- Task precedence relations modelled by a weighted Directed Acyclic Graph (DAG); arc costs c_{ij} indicate amount of data passed from task i to task j if i is a precedent of j
- Processor architecture modelled by a distance matrix; distance between two processors k, l given by d_{kl}
- Delays γ_{ij}^{kl} proportional to $c_{ij} d_{kl}$ if task i is assigned to processor k and precedes task j, assigned to processor $l \neq k$

Simple example

Instance: $D = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$



i	1	2	3	4	5
L_i	2	3	5	8	4

Classic formulation I

- *Indices*: set of tasks V, set of processors P
- *Parameters*: Weighted DAG $G = (V, A, c)$ for task precedences, symmetric distance matrix D for processor architecture, computation times L_i for jobs $i \in V$, $\alpha \gg 0$ sufficiently large penalty coefficient
- *Variables*:

$$
\begin{aligned}
y_{jk}^s &= \begin{cases} 1 & \text{task } j \text{ is } s\text{-th task} \\ & \text{executed on processor } k \\ 0 & \text{otherwise} \end{cases} & \begin{array}{l} \forall\, j, s \in V, \\ \forall\, k \in P \end{array} \\
t_j &= \text{starting time of task } j & \forall\, j \in V
\end{aligned}
$$

Classic formulation II

- Minimize makespan:

$$\min_{y,t} \max_{j \in V} \{t_j + L_j\}$$

- Each task is assigned to only one processor in exactly one slot:

$$\sum_{k \in P} \sum_{s \in V} y_{jk}^{s} = 1 \ \forall j \in V$$

- Each processor has at most one task assigned to the first slot

$$\sum_{j \in V} y_{jk}^{1} \leq 1 \ \forall k \in P$$

- Do not leave any empty slot on any processor

$$\sum_{j \in V} y_{jk}^{s} \leq \sum_{j \in V} y_{jk}^{s-1} \ \forall k \in P, s \in V \backslash \{1\}$$

Classic formulation III

- Starting time of task j depends on starting time of task i if i, j executed on the same processor

$$t_j \geq t_i + L_i - \alpha \left(2 - \left(y_{ik}^s + \sum_{r=s+1}^{|V|} y_{jk}^r \right) \right) \;\; \forall k \in P, s \in V \backslash \{n\}, i, j \in V$$

- Consider communication delays

$$t_j \geq t_i + L_i + \sum_{k,l \in P} \sum_{s,r \in V} \gamma_{ij}^{kl} y_{ik}^s y_{jl}^r \;\; \forall j \in V, i : (i,j) \in A$$

- y binary, $t \geq 0$ real

Usual linearization

- Work in general framework: variables $x_i, i \leq n$ and quadratic terms $x_i x_j, \{i,j\} \in E$
- Substitute $x_i x_j$ with linearization variables w_{ij}, add constraints $w_{ij} = x_i x_j$
- Replace such constraints by the following:

$$\begin{array}{rcl} w_{ij} & \leq & x_i \\ w_{ij} & \leq & x_j \\ w_{ij} & \geq & x_i + x_j - 1 \end{array}$$

- Usual linearization is an exact reformulation
- **Adds $3|E|$ (that is, $O(n^2)$) constraints to the formulation**

Compact linearization

- Multiply assignment constraint $\sum_{i=1}^{n} x_i = 1$ by each x_j
  $\Rightarrow$ get $\sum_{i=1}^{n} x_i x_j = x_j$ ($j \le n$)
- Substitute bilinear terms with w_{ij},
  $\Rightarrow$ get $\sum_{i=1}^{n} w_{ij} = x_j$ ($j \le n$)
- Replace constraints $w_{ij} = x_i x_j$ with:

$$\begin{aligned} w_{ij} &= w_{ji} \quad \forall \{i,j\} \in E \qquad (4) \\ \sum_{i=1}^{n} w_{ij} &= x_j \quad \forall j \le n \qquad (5) \end{aligned}$$

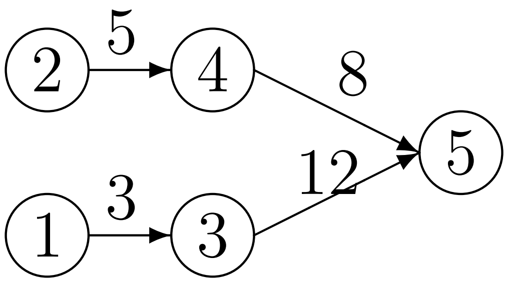
- Constraints $w_{ij} = w_{ji}$ result in variable elimination in presolve stage: **only adds** n **constraints**
- Can be extended to cases with many assignment constraints

Classic formulation IV

- Compact linearization reduces linearization constraints from $|A||V|^2|P|^2$ to $|V|^3|P|$
- Furthermore, it tightens linear relaxation in deepest BB nodes, speeding up convergence (empirical observation)
- Computational savings: 1 to 2 orders of magnitude average
- However, very large CPU timings (over 27 hours for instance with 9 tasks on 3 processors, with compact linearization — 105 hours with usual linearization)
- Experiments carried out on PIV 2.66GHz 1GB RAM
- Limited usefulness

Simple example again

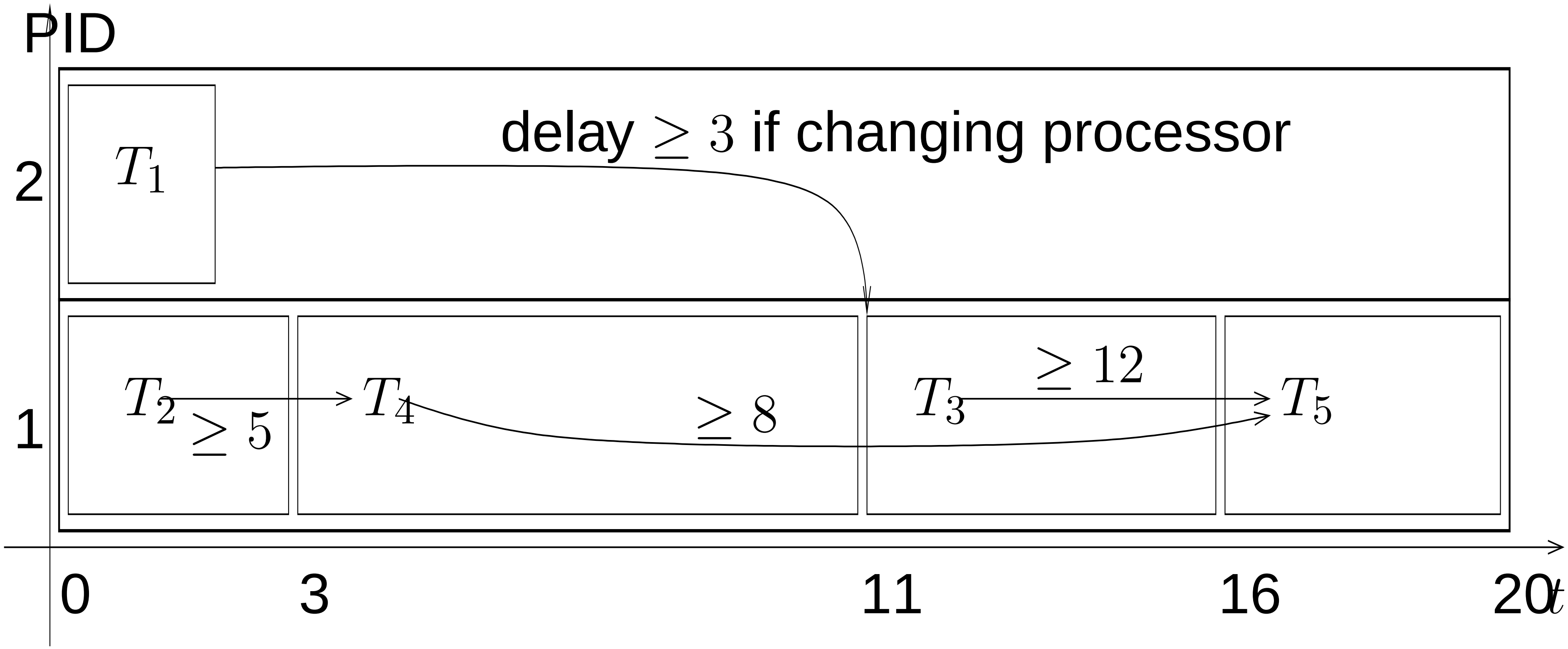
Instance: $D = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$



i	1	2	3	4	5
L_i	2	3	5	8	4

Simple example: solution

Optimal solution: makespan=20



Packing formulation I

- Consider a "big rectangle" $T_{\text{max}} \times |P|$, where T_{max} is an upper bound for the makespan
- Task j represented by a rectangular strip $L_j \times 1$
- Idea: **pack strips into big rectangle**
- We get much more efficient formulation

Packing formulation II

- Indices and parameters as above
- *Variables*:

$$
\begin{aligned}
t_j \geq 0 &: \text{ starting time of task } j \forall\ j \in V \\
y_j \in \mathbb{Z}_+ &: \text{ processor ID where task } j \text{ is executed } \forall\ j \in V \\
z_{jk} \in \{0,1\} &: \left\{ \begin{array}{ll} 1 & \text{task } j \text{ assigned to proc. } k \\ 0 & \text{otherwise} \end{array} \right\} \begin{array}{l} \forall\ j \in V, \\ \forall\ k \in P \end{array} \\
\sigma_{ij} \in \{0,1\} &: \left\{ \begin{array}{ll} 1 & \text{task } i \text{ finishes before } j \text{ starts} \\ 0 & \text{otherwise} \end{array} \right\} \forall\ i,j \in V \\
\epsilon_{ij} \in \{0,1\} &: \left\{ \begin{array}{ll} 1 & \text{PID of task } i < \text{PID of task } j \\ 0 & \text{otherwise} \end{array} \right\} \forall\ i,j \in V
\end{aligned}
$$

Packing formulation II

- Minimize makespan

$$\min \max_{j \in V} \{t_j + L_j\}$$

- Relative positioning possibilities

$$\begin{aligned} \sigma_{ij} + \sigma_{ji} + \epsilon_{ij} + \epsilon_{ji} &\geq 1 \quad \forall\, i \neq j \in V \\ \sigma_{ij} + \sigma_{ji} \leq 1 \ \wedge\ \epsilon_{ij} + \epsilon_{ji} &\leq 1 \quad \forall\, i \neq j \in V \end{aligned}$$

- Constrain starting times if task i finishes before j starts

$$t_j \geq t_i + L_i - (1 - \sigma_{ij}) T_{\text{max}} \quad \forall\, i \neq j \in V$$

- Constrain PIDs if task j has larger PID than i

$$y_j \geq y_i + 1 - (1 - \epsilon_{ij})|P| \quad \forall\, i \neq j \in V$$

Packing formulation III

- Task precendences

$$\sigma_{ij} = 1 \quad \forall j \in V, i : (i,j) \in A$$

- Assignments and relations between PIDs and assignment variables

$$\sum_{k \in P} z_{ik} = 1 \quad \wedge \quad \sum_{k \in P} k z_{ik} = y_i \quad \forall i \in V$$

- Communication delays

$$t_j \geq t_i + L_i + \sum_{k,l \in P} \gamma_{ij}^{kl} z_{ik} z_{jl} \quad \forall j \in V, i : (i,j) \in A$$

Packing formulation IV

- Variables: from 3 indices to 2 (linearization variables: from 6 indices to 4)
- Expect reduction in CPU time of at least $O(|P|^2)$
- Expect compact linearization to make a difference over usual linearization for dense precedence graphs
- **Average CPU time reduction factor: 5000** (much better than $O(|P|^2)$ in our examples)
- Found solutions to medium-scale MSPCD instances in reasonable time
- Compact linearization is beneficial only for dense precedence graphs

Conclusions and Future work

- Presented two formulations for MSPCD
- Packing formulation much tighter than classic formulation
- Compact linearization always beneficial in classic formulation, but only beneficial for dense graphs in packing formulation
- Formulation-based solution approach feasible for medium-scale MSPCD instances
- Ongoing and future work: additional valid cuts and combinatorial branch-and-bound