Lower and upper bounds for the resource-constrained modulo scheduling problem

Christian Artigues¹  Maria Alejandra Ayala²  Abir Benabid³  Claire Hanen⁴

¹LAAS - CNRS & Université de Toulouse, France
²Universidad de los Andes, Mérida, Venezuela
³King Saud University, Saudi Arabia
⁴LIP6 & Université Paris Ouest Nanterre, France

artigues@laas.fr, marialej@ula.ve, ben_abid_abir@yahoo.fr, Claire.Hanen@lip6.fr

PMS 2012 - Leuven
Outline

1. Problem definition
2. Typical application: instruction scheduling for VLIW processors
3. Solution methods
4. Computational experiments
Outline

1 Problem definition

2 Typical application: instruction scheduling for VLIW processors

3 Solution methods
   - Integer Linear programming (ILP) for the RCMSP
   - Decomposed Software Pipelining
   - An hybrid method

4 Computational experiments
Periodic scheduling

- Set $V$ of unit-duration tasks with $|V| = n$.
- Each task $i \in V$ has an infinite number of occurrences $<i; q>$ that are scheduled periodically.
- A start time $\sigma_i^q \in \mathbb{N}$ has to be assigned to each task occurrence $<i; q>$ such that
  \[ \sigma_i^q = \sigma_i^0 + q\lambda \]
  where $\lambda$ is the period (to be minimized).

A periodic schedule is defined by $\sigma_i \equiv \sigma^0$, $\forall i$ with
$\sigma_i \in \{0, \ldots, \lambda - 1\}$
Uniform precedence constraints

- Set $E$ of precedence constraints such that $(i, j) \in E$ is defined by a latency $\theta_i^j$ and a distance $\omega_i^j$

$$\sigma_j^q + \omega_i^j \geq \sigma_i^q + \theta_i^j, \quad \forall (i, j) \in E, \forall q \in \mathbb{N}$$

\[
\begin{array}{c|c|c|c|c}
<i;q> & <i;q+1> & <i;q+2> & <i;q+3> \\
\hline
<j;q> & <j;q+1> & <j;q+2> & <j;q+3> \\
\hline
<k;q> & <k;q+1> & <k;q+2> & <k;q+3> \\
\hline
\end{array}
\]

$\lambda = 1$
Uniform precedence constraints

Set $E$ of precedence constraints such that $(i, j) \in E$ is defined by a latency $\theta^j_i$ and a distance $\omega^j_i$

$$\sigma^q_{j} + \omega^j_i \geq \sigma^q_i + \theta^j_i, \quad \forall (i, j) \in E, \forall q \in \mathbb{N}$$

\[ \begin{array}{cccc}
<i;q> & <i;q+1> & <i;q+2> & <i;q+3> \\
0 & 0 & 0 & 0 \\
<j;q> & <j;q+1> & <j;q+2> & <j;q+3> \\
<k;q> & <k;q+1> & <k;q+2> & <k;q+3> \\
\end{array} \]

$\lambda = 1$
Uniform precedence constraints

Set $E$ of precedence constraints such that $(i, j) \in E$ is defined by a latency $\theta_j^i$ and a distance $\omega_j^i$.

$$\sigma_j^q + \omega_j^i \geq \sigma_i^q + \theta_i^j, \quad \forall (i, j) \in E, \forall q \in \mathbb{N}$$

$\lambda = 1$
Set $E$ of precedence constraints such that $(i,j) \in E$ is defined by a latency $\theta^j_i$ and a distance $\omega^j_i$

$$\sigma^q_j + \omega^j_i \geq \sigma^q_i + \theta^j_i, \quad \forall (i,j) \in E, \forall q \in \mathbb{N}$$

$\lambda = 1$
Uniform precedence constraints

Set $E$ of precedence constraints such that $(i, j) \in E$ is defined by a latency $\theta^{j}_{i}$ and a distance $\omega^{j}_{i}$

$$\sigma^{q}_{j} + \omega^{j}_{i} \geq \sigma^{q}_{i} + \theta^{j}_{i}, \quad \forall (i, j) \in E, \forall q \in \mathbb{N}$$

Artigues, Ayala, Benabid, Hanen

$\lambda = 1$
Uniform precedence constraints

- Set $E$ of precedence constraints such that $(i, j) \in E$ is defined by a latency $\theta^j_i$ and a distance $\omega^j_i$

$$\sigma^q_j + \omega^j_i \geq \sigma^q_i + \theta^j_i, \quad \forall (i, j) \in E, \forall q \in \mathbb{N}$$

\[\begin{array}{cccc}
<i;0> & <i;1> & <i;2> & <i;3> \\
<j;0> & <j;1> & <j;2> & <j;3> \\
<k;0> & <k;1> & <k;2> & <k;3> \\
& & & \\
& & & \\
\end{array}\]

\[\begin{array}{ccc}
1, 1 & 0, 0 & 2, 1 \\
1, 2 & & \\
\end{array}\]

$$\lambda = 1$$
Uniform precedence constraints

- Set $E$ of precedence constraints such that $(i, j) \in E$ is defined by a latency $\theta^j_i$ and a distance $\omega^j_i$.

$$\sigma^q_j + \omega^j_i \geq \sigma^q_i + \theta^j_i, \quad \forall (i, j) \in E, \forall q \in \mathbb{N}$$

\[ \begin{array}{cccc}
<i;q> & <i;q+1> & <i;q+2> & <i;q+3> \\
<j;q> & <j;q+1> & <j;q+2> & <j;q+3> \\
<k;q-1> & <k;q> & <k;q+1> & <k;q+2> & <k;q+3> \\
\end{array} \]

- $\lambda = 1$
Uniform precedence constraints

- Set $E$ of precedence constraints such that $(i, j) \in E$ is defined by a latency $\theta^j_i$ and a distance $\omega^j_i$

$$\sigma^q_j + \omega^j_i \geq \sigma^q_i + \theta^j_i, \quad \forall (i, j) \in E, \forall q \in \mathbb{N}$$

$\lambda = 1$
Uniform precedence constraints

Set $E$ of precedence constraints such that $(i, j) \in E$ is defined by a latency $\theta^j_i$ and a distance $\omega^j_i$

$$\sigma^q_j + \omega^j_i \geq \sigma^q_i + \theta^j_i, \quad \forall (i, j) \in E, \forall q \in \mathbb{N}$$

$\lambda = 1$
Uniform precedence constraints

Set $E$ of precedence constraints such that $(i, j) \in E$ is defined by a latency $\theta^j_i$ and a distance $\omega^j_i$

$$\sigma^q_j + \omega^j_i \geq \sigma^q_i + \theta^j_i, \quad \forall (i, j) \in E, \forall q \in \mathbb{N}$$

$\lambda = 1$
Set $E$ of precedence constraints such that $(i, j) \in E$ is defined by a latency $\theta^j_i$ and a distance $\omega^j_i$.

$$\sigma^q_j + \omega^j_i \geq \sigma^q_i + \theta^j_i, \quad \forall (i, j) \in E, \forall q \in \mathbb{N}$$

$\lambda = 2$
Uniform precedence constraints

Set $E$ of precedence constraints such that $(i, j) \in E$ is defined by a latency $\theta_j^i$ and a distance $\omega_j^i$

$$\sigma_j^{q+\omega_j^i} \geq \sigma_i^q + \theta_j^i, \quad \forall (i, j) \in E, \forall q \in \mathbb{N}$$

$\lambda = 2$, pattern $\sigma_i = 1$, $\sigma_j = 0$, $\sigma_k = 3$, $C_{\text{max}} = 4$
Basic Cyclic scheduling problem

Precedence constraints can be expressed using only $\sigma_i$:

$$\sigma_j^{q+\omega_i^j} \geq \sigma_i^q + \theta_i^j \quad \Leftrightarrow \quad \sigma_j + \lambda(q + \omega_i^j) \geq \sigma_i + \lambda q + \theta_i^j$$

$$\Leftrightarrow \quad \sigma_j \geq \sigma_i + \theta_i^j - \lambda \omega_i^j$$

The Basic Cyclic Scheduling Problem (BCSP)

$$\min \lambda$$

$$\sigma_j \geq \sigma_i + \theta_i^j - \lambda \omega_i^j$$

$$\forall (i,j) \in E$$

$$\sigma_i \in \mathbb{N}$$

$$\forall i \in \{1, \ldots, n\}$$

Remark: for a fixed $\lambda$ we obtain a project scheduling problem with minimum and maximum time lags that can be solved by the Bellman-Ford algorithm.
Solving the BCSP

The BCSP is polynomial [Chrétienne 85], [Hanen and Munier 95]

**Necessary and sufficient feasibility condition**

There exists a feasible schedule if and only if for any circuit $C$ of $G(V, E)$, $\omega(C) \leq 0 \implies \theta(C) \leq 0$

where $\theta(C) = \sum_{(i,j) \in C} \theta^i_j$ and $\omega(C) = \sum_{(i,j) \in C} \omega^i_j$.

**Computation of the optimal period**

Critical circuit $C^* = \arg\max_C \theta(C)/\omega(C)$. The optimal period is the smallest integer $\lambda$ such that $\lambda \geq \theta(C^*)/\omega(C^*)$.

![Diagram of a circuit]
Resource constraints

- A set of $m$ resources
- Each resource $s$ has a limited availability $B_s$
- Each task requires a non-negative amount $b_{is}$ of each resource
- At each time point, each resource cannot be oversubscribed.

Example: a single resource $s = 1$, $B_s = 3$, $b_{is} = 2$, $b_{js} = 1$, $b_{ks} = 1$

\[
\begin{array}{|c|c|c|c|}
\hline
<i; q> & <i; q + 1> & <i; q + 2> & <i; q + 3> \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|c|}
\hline
<j; q> & <j; q + 1> & <j; q + 2> & <j; q + 3> \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|c|c|}
\hline
<k; q - 1> & <k; q> & <k; q + 1> & <k; q + 2> & <k; q + 3> \\
\hline
\end{array}
\]

\[
\begin{array}{cccc}
4 & 4 & 4 & 4 \\
& & & \hat{N}
\end{array}
\]

$\lambda = 1$
Resource constraints

- A set of $m$ resources
- Each resource $s$ has a limited availability $B_s$
- Each task requires a non-negative amount $b_{is}$ of each resource
- At each time point, each resource cannot be oversubscribed.

Example: a single resource $s = 1$, $B_s = 3$, $b_{is} = 2$, $b_{js} = 1$, $b_{ks} = 1$

\[
\begin{align*}
&\langle i; q-1 \rangle &\langle i; q \rangle &\langle i; q+1 \rangle &\langle i; q+2 \rangle \\
&\langle j; q \rangle &\langle j; q+1 \rangle &\langle j; q+2 \rangle &\langle j; q+3 \rangle \\
&\langle k; q-2 \rangle &\langle k; q-1 \rangle &\langle k; q \rangle &\langle k; q+1 \rangle &\langle k; q+2 \rangle \\
\end{align*}
\]

$\lambda = 2$
The Resource-constrained modulo scheduling Problem

\[
\begin{align*}
\min \lambda \\
\sigma_j & \geq \sigma_i + \theta_i^j - \lambda \omega_i^j & \forall (i,j) \in E \\
\sum_{i \in V | \sigma_i \mod \lambda = \tau} b_i^s & \leq B_s & \forall \tau \in \{0, \ldots, \lambda - 1\}, \forall s \in \{1, \ldots, m\} \\
\sigma_i & \in \mathbb{N} & \forall i \in \{1, \ldots, n\}
\end{align*}
\]

- The RCMSP is strongly NP-hard.
- Precedence lower bound \([\theta(C^*)/\omega(C^*)]\)
- Resource lower bound \(\max_{s=1,\ldots,m} \sum_{i \in V} b_i s / B_s\)
Outline

1. Problem definition

2. Typical application: instruction scheduling for VLIW processors

3. Solution methods
   - Integer Linear programming (ILP) for the RCMSP
   - Decomposed Software Pipelining
   - An hybrid method

4. Computational experiments
Typical application: instruction scheduling for VLIW processors

```
int prod(int n, short a[], short b) {
    int s=0, i;
    for (i=0;i<n;i++) {
        s += a[i]*b;
    }
    return s;
}
```

L?__0_8:
LDH_1 g131 = 0, G127
MULL_2 g132 = G126, g131
ADD_3 G129 = G129, g132
ADD_4 G128 = G128, 1
ADD_5 G127 = G127, 2
CMPNE_6 b135 = G118, G128
BRF_7 b135, L?__0_8

Schedule instructions to end the program in minimum time.

Optimize loops performance.

Software pipeline

Cyclic scheduling problem

Modulo scheduling
Typical application: resources and solution

Exemple.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>4</td>
</tr>
<tr>
<td>MEM</td>
<td>1</td>
</tr>
<tr>
<td>CTL</td>
<td>1</td>
</tr>
<tr>
<td>ODD</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RESERVATION</th>
<th>ALU</th>
<th>MEM</th>
<th>CTL</th>
<th>ODD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ALUX</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MUL</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MULX</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MEM</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MEMX</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>CTL</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ \lambda = 2 \]
Outline

1. Problem definition

2. Typical application: instruction scheduling for VLIW processors

3. Solution methods
   - Integer Linear programming (ILP) for the RCMSP
   - Decomposed Software Pipelining
   - An hybrid method

4. Computational experiments
Solution methods

- Decomposed Software pipelining (DSP) [Darte et al 00] [Gasperoni et schwiegelshohn 94] [Benabid and Hanen 11]
- Integer Linear Programming (ILP) [Eichenberger and Davidson 97] [Dupont De Dinechin 05] [Ayala and A. 11]
- Constraint Programming (CP) [Bonfietti et al 11]

In this talk:

→ A new hybrid method based on DSP and ILP

→ Comparison of solution methods (DSP, CP, ILP, hybrid) and of a column generation-based lower bound (CG) [Ayala and A. 11] on a set of mixed industrial/randomly generated instances
Outline

1. Problem definition
2. Typical application: instruction scheduling for VLIW processors
3. Solution methods
   - Integer Linear programming (ILP) for the RCMSP
   - Decomposed Software Pipelining
   - An hybrid method
4. Computational experiments
ILP for the RCMSP

Linearization is easier with a fixed $\lambda$. The optimum value is computed by successive ILP solving with a fixed $\lambda$.

- **Direct formulation** [Dupont De Dinechin 05]
  - Start time $\sigma_i \in [0, T - 1]$ modeled by a binary variable $x_{it}$, $i \in V$, $t \in [0, T - 1]$ where $T$ is an upper bound on the makespan of an optimal schedule.

- **Decomposed formulation** [Eichenberger and Davidson 97]
  \[
  \sigma_i = \tau_i + \lambda k_i
  \]
  - $k_i \in \{0, \ldots, \left\lfloor \frac{T - 1}{\lambda} \right\rfloor\}$, $k_i$ is the iteration in which task $i$ is placed. Modeled by an integer variable.
  - $\tau_i = \sigma_i \mod \lambda$, is the start time of operation $i$ in the interval $\{0, \ldots, \lambda - 1\}$. Modeled by a binary variable $z_{i\tau}$, $i \in V$, $\tau \in \{0, \ldots, \lambda - 1\}$
The two formulations yield the same lower bound by solving successively their LP relaxations [Ayala and A. 11].
Decomposed formulation (EF) (Eichenberger et al 1997)
Binary variables $z^T_i$ such that, $\tau_i = \sum_{\tau=0}^{\lambda-1} \tau z^T_i$, $i = 1, ..., n$.

$$
\min \sum_{i=1}^{n} w_i \left( \sum_{\tau=0}^{\lambda-1} \tau z^T_i + k_i \lambda \right)
$$

$$
\sum_{\tau=0}^{\lambda-1} z^T_i = 1, \forall i \in [1, n]
$$

$$
\sum_{\tau=0}^{\lambda-1} \tau z^T_i + k_i \lambda + \theta_i^j - \lambda \omega_i^j \leq \sum_{\tau=0}^{\lambda-1} \tau z^T_j + k_j \lambda, \forall (i, j) \in E
$$

$$
\sum_{i=1}^{n} z^T_i b^s_i \leq B_s, \forall s \in m, \tau \in [0, \lambda)
$$

$z^T_i \in \{0, 1\}, \forall i = 1, ..., n, \forall \tau \in [0, \lambda - 1]$

$k_i \in \mathbb{N}, \forall i = 1, ..., n$
Outline

1. Problem definition

2. Typical application: instruction scheduling for VLIW processors

3. Solution methods
   - Integer Linear programming (ILP) for the RCMSP
   - Decomposed Software Pipelining
   - An hybrid method

4. Computational experiments
Decomposed Software Pipelining (1/2)

DSP is a two-phase heuristic

- **First phase**: find a legal *retiming* of the tasks

  \[ R : V \rightarrow \mathbb{N}, \forall (i, j) \in E, R_j + \omega_i^j - R_i \geq 0 \]

- **Second phase**:
  - Build a new graph \( G^R \) keeping only arcs \((i, j) \in E\) such that
    \[ R_j + \omega_i^j - R_i = 0 \]
  - Schedule the tasks with an acyclic list scheduling algorithm satisfying the resource constraints and the precedence constraints (considering only \( \theta_i^j \)) in \( G^R \). Let \( \pi_i \); the start time of \( i \in V \) in this schedule

**Theorem [Benabid and Hanen 11]**

Setting \( \sigma_i = \pi_i + R_i \lambda^R \) yields a feasible schedule of period \( \lambda^R \) for the RCMP where \( \lambda^R = \max \left( \max_{i \in V} \pi_i + 1, \max_{(i, j) \in G \backslash G^R} \frac{\pi_i - \pi_j + \theta_i^j}{R_j - R_i + \omega_i^j} \right) \)
Decomposed Software Pipelining (2/2)

How to choose a retiming?

- [Gasperoni et al. 94]
  Compute the resource-unconstrained optimal schedule $\sigma^\infty$ and the corresponding optimal period $\lambda^\infty$ and set
  \[ R_i = \lfloor \sigma^\infty / \lambda^\infty \rfloor, \forall i \in V \]

- [Darte and Huard 00]
  Compute a retiming that minimizes the number of arcs in $G^R$ through a min cost flow computation.

List Scheduling Algorithm?

- Define a list of tasks compatible with the precedence constraints in $G^R$

- Schedule the tasks as early as possible following the order of the list.

→ DSP has worst-case performance guarantee [Benabid and Hanen 11].
Outline

1. Problem definition

2. Typical application: instruction scheduling for VLIW processors

3. Solution methods
   - Integer Linear programming (ILP) for the RCMSP
   - Decomposed Software Pipelining
   - An hybrid method

4. Computational experiments
An hybrid method

Merits and drawbacks of the existing methods:

- ILP formulations define an exact solving scheme but are too large to solve practical problems to optimality or even to find a feasible solution in reasonable CPU time.
- DSP algorithms are very fast, with guaranteed performance but may yield suboptimal schedules.

The proposed hybrid methods aim at improving the solutions found by DSP algorithms by replacing the list scheduling algorithm by exact solving through ILP. This is done by:

- Setting $k_i \leftarrow R_i$, $\forall i \in V$
- Solving the Eihenberger and Davidson formulation with variables $z_{i\tau}$ only.
Outline

1. Problem definition

2. Typical application: instruction scheduling for VLIW processors

3. Solution methods
   - Integer Linear programming (ILP) for the RCMSP
   - Decomposed Software Pipelining
   - An hybrid method

4. Computational experiments
Benchmark instances

- 43 instances from the ST200 compiler.
- To make these instances harder to solve, the task demand on each of the 6 resources was randomly generated between 0 and 10 and the capacity of each resource was set to 10.
- Smallest instance: 10 operations and 42 precedence constraints, largest instance 214 operations and 1063 precedence constraints.

Comparison of DSP, ILP, hybrid with
- the precedence-based and resource-based trivial lower bound.
- the CP method by [Bonfietti et al. 11]: tackle directly the modulo-constraints without the need of fixing the period value.
- the CG lower bound by [Ayala and A. 11]: based on the Dantzig-Wolfe decomposition of the ILP models (resource constraints).
## Computational results

<table>
<thead>
<tr>
<th>Instances</th>
<th>$n$</th>
<th>DSP $\lambda_{DSP}$</th>
<th>hybrid/HD $\lambda_{hyb}$</th>
<th>CPU $s$</th>
<th>hybrid/GS $\lambda_{hyb}$</th>
<th>CPU $s$</th>
<th>CP $\lambda_{CP}$</th>
<th>ILP+ ($P_+^{ILP}$) $\lambda_{ILP+}$</th>
<th>CPU $s$</th>
<th>CG (DW$^+$) $\lambda_{CG}$</th>
<th>CPU $s$</th>
<th>$\lambda^0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>adpcm-st231.1</td>
<td>86</td>
<td>80</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>80</td>
<td>-</td>
<td>-</td>
<td>55</td>
<td>301</td>
<td>52</td>
</tr>
<tr>
<td>adpcm-st231.2</td>
<td>142</td>
<td>139</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>139</td>
<td>-</td>
<td>-</td>
<td>82</td>
<td>305</td>
<td>82</td>
</tr>
<tr>
<td>gsm-st231.1</td>
<td>30</td>
<td>30</td>
<td>29</td>
<td>2</td>
<td>28</td>
<td>2</td>
<td>28</td>
<td>28*</td>
<td>256</td>
<td>25</td>
<td>8</td>
<td>24</td>
</tr>
<tr>
<td>gsm-st231.2</td>
<td>101</td>
<td>93</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>93</td>
<td>-</td>
<td>-</td>
<td>61</td>
<td>301</td>
<td>59</td>
</tr>
<tr>
<td>gsm-st231.5</td>
<td>44</td>
<td>36</td>
<td>36</td>
<td>10</td>
<td>36</td>
<td>17</td>
<td>36*</td>
<td>3343</td>
<td>36</td>
<td>37</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>gsm-st231.6</td>
<td>30</td>
<td>27</td>
<td>27</td>
<td>3</td>
<td>27</td>
<td>4</td>
<td>27</td>
<td>27*</td>
<td>3</td>
<td>3</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>gsm-st231.7</td>
<td>44</td>
<td>41</td>
<td>41</td>
<td>13</td>
<td>41</td>
<td>17</td>
<td>41</td>
<td>256</td>
<td>41</td>
<td>66</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>gsm-st231.8</td>
<td>14</td>
<td>12</td>
<td>12</td>
<td>0.3</td>
<td>12</td>
<td>0.3</td>
<td>12</td>
<td>0.6</td>
<td>12</td>
<td>&lt;0.1</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>gsm-st231.9</td>
<td>34</td>
<td>32</td>
<td>32</td>
<td>2</td>
<td>34</td>
<td>4</td>
<td>32</td>
<td>62</td>
<td>31</td>
<td>12</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>gsm-st231.10</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>0.2</td>
<td>8</td>
<td>0.1</td>
<td>8*</td>
<td>0.2</td>
<td>8</td>
<td>&lt;0.1</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>gsm-st231.11</td>
<td>26</td>
<td>24</td>
<td>24</td>
<td>1</td>
<td>24</td>
<td>1</td>
<td>24</td>
<td>5</td>
<td>24</td>
<td>1.4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>gsm-st231.12</td>
<td>15</td>
<td>13</td>
<td>13</td>
<td>0.3</td>
<td>13</td>
<td>0.4</td>
<td>13</td>
<td>0.7</td>
<td>13</td>
<td>&lt;0.1</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>gsm-st231.13</td>
<td>46</td>
<td>43</td>
<td>43</td>
<td>168</td>
<td>42</td>
<td>440</td>
<td>43</td>
<td>11265</td>
<td>41</td>
<td>125</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>gsm-st231.14</td>
<td>39</td>
<td>34</td>
<td>34</td>
<td>6</td>
<td>34</td>
<td>10</td>
<td>34</td>
<td>3766</td>
<td>33</td>
<td>17</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>gsm-st231.15</td>
<td>15</td>
<td>12</td>
<td>12</td>
<td>0.3</td>
<td>12</td>
<td>0.3</td>
<td>12</td>
<td>0.9</td>
<td>12</td>
<td>&lt;0.1</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>gsm-st231.16</td>
<td>65</td>
<td>59</td>
<td>59</td>
<td>145</td>
<td>59</td>
<td>144</td>
<td>59</td>
<td>8666</td>
<td>48</td>
<td>300</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td>gsm-st231.17</td>
<td>38</td>
<td>33</td>
<td>33</td>
<td>202</td>
<td>-</td>
<td>-</td>
<td>33</td>
<td>12786</td>
<td>33</td>
<td>19</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>gsm-st231.18</td>
<td>214</td>
<td>194</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>193</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>120</td>
</tr>
<tr>
<td>gsm-st231.19</td>
<td>19</td>
<td>15</td>
<td>15</td>
<td>0.4</td>
<td>15</td>
<td>0.6</td>
<td>15</td>
<td>1.6</td>
<td>15</td>
<td>0.2</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>gsm-st231.20</td>
<td>23</td>
<td>20</td>
<td>20</td>
<td>1</td>
<td>20</td>
<td>1.4</td>
<td>20</td>
<td>17</td>
<td>20</td>
<td>0.9</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>gsm-st231.21</td>
<td>33</td>
<td>30</td>
<td>30</td>
<td>6</td>
<td>30</td>
<td>6</td>
<td>30</td>
<td>6105</td>
<td>29</td>
<td>7</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>gsm-st231.22</td>
<td>31</td>
<td>29</td>
<td>29</td>
<td>3</td>
<td>29</td>
<td>4</td>
<td>29</td>
<td>51</td>
<td>28</td>
<td>7</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>gsm-st231.25</td>
<td>60</td>
<td>55</td>
<td>-</td>
<td>-</td>
<td>55</td>
<td>75</td>
<td>57</td>
<td>11589</td>
<td>48</td>
<td>300</td>
<td>37</td>
<td>37</td>
</tr>
<tr>
<td>gsm-st231.29</td>
<td>44</td>
<td>42</td>
<td>42</td>
<td>13</td>
<td>42</td>
<td>15</td>
<td>42</td>
<td>63</td>
<td>42</td>
<td>68</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>gsm-st231.30</td>
<td>30</td>
<td>25</td>
<td>25</td>
<td>3</td>
<td>25</td>
<td>6</td>
<td>25</td>
<td>14</td>
<td>25</td>
<td>6</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>gsm-st231.31</td>
<td>44</td>
<td>39</td>
<td>39</td>
<td>13</td>
<td>39</td>
<td>17</td>
<td>39</td>
<td>833</td>
<td>39</td>
<td>59</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>gsm-st231.32</td>
<td>32</td>
<td>30</td>
<td>30</td>
<td>4</td>
<td>30</td>
<td>5</td>
<td>30</td>
<td>11</td>
<td>30</td>
<td>5</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>gsm-st231.33</td>
<td>59</td>
<td>52</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>46</td>
<td>9697</td>
<td>46</td>
<td>300</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td>gsm-st231.34</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>&lt;0.1</td>
<td>7</td>
<td>0.1</td>
<td>7*</td>
<td>0.2</td>
<td>7</td>
<td>&lt;0.1</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>gsm-st231.35</td>
<td>18</td>
<td>16</td>
<td>14</td>
<td>0.4</td>
<td>14</td>
<td>0.5</td>
<td>14</td>
<td>4</td>
<td>14</td>
<td>0.2</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>gsm-st231.36</td>
<td>31</td>
<td>29</td>
<td>24</td>
<td>2</td>
<td>24</td>
<td>10</td>
<td>24</td>
<td>321</td>
<td>24</td>
<td>4</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>gsm-st231.39</td>
<td>26</td>
<td>23</td>
<td>21</td>
<td>1.5</td>
<td>21</td>
<td>3</td>
<td>21</td>
<td>376</td>
<td>20</td>
<td>2</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>gsm-st231.40</td>
<td>21</td>
<td>17</td>
<td>16</td>
<td>0.5</td>
<td>18</td>
<td>1.3</td>
<td>17</td>
<td>28</td>
<td>16</td>
<td>0.6</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>gsm-st231.41</td>
<td>60</td>
<td>50</td>
<td>-</td>
<td>-</td>
<td>47</td>
<td>286</td>
<td>49</td>
<td>10069</td>
<td>46</td>
<td>300</td>
<td>34</td>
<td>34</td>
</tr>
<tr>
<td>gsm-st231.42</td>
<td>23</td>
<td>19</td>
<td>18</td>
<td>0.7</td>
<td>19</td>
<td>4</td>
<td>18</td>
<td>20</td>
<td>18</td>
<td>1</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>gsm-st231.43</td>
<td>26</td>
<td>23</td>
<td>21</td>
<td>2</td>
<td>22</td>
<td>2</td>
<td>20</td>
<td>101</td>
<td>20</td>
<td>2</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

#best(opt) | 23  | 25  | 25  | 27(2) | 27(27)
Result analysis

- ILP is non-dominated by CP (better for proving optimality, lower bounds are obtained)
- CG lower bound competitive with the ILP-based bounds and significantly better than the trivial bound (while the LP relaxation-based bounds never exceeds the trivial bound)
- DSP: really fast, reasonable solutions but can be far from optimum
- Hybrid method significantly improves DSP while requiring much less CPU time than ILP
- Hybrid method outperforms CP on 8 instances. (the reverse holds for 1 instance when both find a solution)
- CP finds solutions on 6 instances where hybrid does find a solution (the reverse holds for 1 instance)
Further research

- Column generation-based heuristics / Branch-and-price
- Hybrid CP/ILP or CS/DSP method
- Dedicated branch and bound
- $k$- periodic schedules?