Integer Linear Programs for Synthesis Problems

SS 2009
Hw/Sw Codesign

Christian Plessl
Overview

- Integer Linear Programs (ILP) basics
- ILP for the partitioning problem (system synthesis)
- ILP for architecture synthesis (hardware synthesis)
ILP Basics (1)

- **Linear Program (LP)**
  - is a special kind of optimization problem
  
  \[
  \begin{align*}
  A \cdot x & \leq b \\
  x & \geq 0 \\
  \max c^T \cdot x
  \end{align*}
  \]

  - real variables
  - linear constraints modeled by (in)equations
  - linear cost function

- **Integer Linear Program (ILP)**
  - an LP where all variables must be integer

  \[
  x \in \mathbb{Z}^n
  \]

- **0-1 LP (binary LP, ZOLP)**
  - an LP where all variables must be binary

  \[
  x \in \{0,1\}^n
  \]
ILP Basics (2)

- Example
  - 3 binary variables $x_1$, $x_2$, $x_3$
  - constraint that at least two variables must be set
  - cost function

- Solution

$$x_1, x_2, x_3 \in \{0,1\}$$

$$x_1 + x_2 + x_3 \geq 2$$

$$\min 5x_1 + 6x_2 + 4x_3$$

<table>
<thead>
<tr>
<th>x_1</th>
<th>x_2</th>
<th>x_3</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>15</td>
</tr>
</tbody>
</table>
ILP Basics (3)

• Solving ILPs
  – ILPs are NP-complete
  – in the worst-case exponential runtime
  – solved by branch&bound algorithms to optimality
    (runtime depends on the efficiency of the model)
  – solvers are eg. lp_solve (http://lpsolve.sourceforge.net/5.5/)
    or cplex (http://www.ilog.com/products/cplex/)

• Challenges
  – translate the problem into an efficient ILP model
  – use only linear constraints and cost function
    ▪ non-linearities of the problem can (possibly) be modeled by several
      linear constraints, but this increases the complexity of the ILP
Overview

- Integer Linear Programs (ILP) basics
- ILP for the partitioning problem (system synthesis)
- ILP for architecture synthesis (hardware synthesis)
Example: Hw/Sw Bi-Partitioning (1)

Problem

1
2
3
4
5
6

Architecture

processor

bus

ASIC

cost table
(all bindings possible)

<table>
<thead>
<tr>
<th>task</th>
<th>SW cost</th>
<th>HW cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>80</td>
<td>320</td>
</tr>
<tr>
<td>2</td>
<td>240</td>
<td>170</td>
</tr>
<tr>
<td>3</td>
<td>710</td>
<td>120</td>
</tr>
<tr>
<td>4</td>
<td>130</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>400</td>
</tr>
<tr>
<td>6</td>
<td>80</td>
<td>260</td>
</tr>
</tbody>
</table>

→ find optimal allocation & binding
ILP for Partitioning

- Binary variables $x_{i,k} = 1$: object $o_i$ in block $p_k$
- Cost $c_{i,k}$, if object $o_i$ is in block $p_k$
- Integer linear program:

$$x_{i,k} \in \{0,1\}, \quad 1 \leq i \leq n, 1 \leq k \leq m$$

$$\sum_{k=1}^{m} x_{i,k} = 1, \quad 1 \leq i \leq n$$

minimize

$$\sum_{k=1}^{m} \sum_{i=1}^{n} x_{i,k} \cdot c_{i,k}, \quad 1 \leq k \leq m, 1 \leq i \leq n$$
**Example: Hw/Sw Bi-Partitioning (2)**

specification graph (incomplete)

```
1
\[ x_{11} \]
\[ x_{12} \]
\[ x_{21} \]
\[ x_{22} \]
2
3
4
5
6
```

\( n \times m = 12 \)
\( \text{n=6} \)
\( \text{m=2} \)

binary variables

```
\begin{array}{c|c|c}
\text{task} & \text{SW cost} & \text{HW cost} \\
1 & 80 & 320 \\
2 & 240 & 170 \\
3 & 710 & 120 \\
4 & 130 & 20 \\
5 & 100 & 400 \\
6 & 80 & 260 \\
\end{array}
```
Example: Hw/Sw Bi-Partitioning (3)

/****************************/
/* objective function */
/****************************/

/*
c1s = 80  c1h = 320 */
/* c2s = 240  c2h = 170 */
/* c3s = 710  c3h = 120 */
/* c4s = 130  c4h = 20  */
/* c5s = 100  c5h = 400 */
/* c6s = 80   c6h = 260 */
*/

min: 80 x11 + 320 x12 + 240 x21 + 170 x22 + 710 x31 + 120 x32 + 130 x41 + 20 x42 + 100 x51 + 400 x52 + 80 x61 + 260 x62;

/****************************/
/* unique mapping constraints */
/****************************/

x11 + x12 = 1;
x21 + x22 = 1;
x31 + x32 = 1;
x41 + x42 = 1;
x51 + x52 = 1;
x61 + x62 = 1;

/****************************/
/* variables in {0,1} */
/****************************/

x11 <= 1;
x12 <= 1;
x21 <= 1;
x22 <= 1;
x31 <= 1;
x32 <= 1;
x41 <= 1;
x42 <= 1;
x51 <= 1;
x52 <= 1;
x61 <= 1;
x62 <= 1;

/****************************/
/* integer variables */
/****************************/

int x11;
int x12;
int x21;
int x22;
int x31;
int x32;
int x41;
int x42;
int x51;
int x52;
int x61;
int x62;
Example: Hw/Sw Bi-Partitioning (4)

Allocation & binding

Cost table

<table>
<thead>
<tr>
<th>task</th>
<th>SW cost</th>
<th>HW cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>80</td>
<td>320</td>
</tr>
<tr>
<td>2</td>
<td>240</td>
<td>170</td>
</tr>
<tr>
<td>3</td>
<td>710</td>
<td>120</td>
</tr>
<tr>
<td>4</td>
<td>130</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>400</td>
</tr>
<tr>
<td>6</td>
<td>80</td>
<td>260</td>
</tr>
</tbody>
</table>

total cost = 570

n * m = 12 binary variables

n = 6
m = 2

V_{processor}

V_{ASIC}
ILP for Partitioning - Cost Constraints

- Maximal number of $h_k$ objects in block $p_k$

$$\sum_{i=1}^{n} x_{i,k} \leq h_k \quad 1 \leq k \leq m$$

- Maximal cost $H_k$ of objects in block $p_k$

$$\sum_{i=1}^{n} c_{i,k} \cdot x_{i,k} \leq H_k \quad 1 \leq k \leq m$$
Example: Hw/Sw Bi-Partitioning (5)

- Constraint on the hardware cost
  - cost of all tasks mapped to hardware must not exceed 300

\[
\sum_{i=1}^{6} c_{i,2} \cdot x_{i,2} \leq 300
\]

```
/**************************/
/* hardware cost constraint */
/**************************/
320 *x12 + 170 *x22 + 120 *x32 + 20 *x42 + 400 *x52 + 260 *x62 <= 300;
```
Example: Hw/Sw Bi-Partitioning (6)

allocation & binding

<table>
<thead>
<tr>
<th>task</th>
<th>SW cost</th>
<th>HW cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>80</td>
<td>320</td>
</tr>
<tr>
<td>2</td>
<td>240</td>
<td>170</td>
</tr>
<tr>
<td>3</td>
<td>710</td>
<td>120</td>
</tr>
<tr>
<td>4</td>
<td>130</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>400</td>
</tr>
<tr>
<td>6</td>
<td>80</td>
<td>260</td>
</tr>
</tbody>
</table>

cost table

constraint: HW cost \(\leq 300\)

total cost = 640
Example: Hw/Sw Bi-Partitioning (7)

- Including communication
  - assume local communication is for free (SW-SW or HW-HW) but bus communication incurs costs (SW-HW or HW-SW)
  (for simplicity, we assume additive costs)

<table>
<thead>
<tr>
<th>edge $i \rightarrow j$</th>
<th>local communication</th>
<th>bus communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>1→2</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>1→3</td>
<td>0</td>
<td>250</td>
</tr>
<tr>
<td>2→4</td>
<td>0</td>
<td>200</td>
</tr>
<tr>
<td>2→5</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>3→5</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>4→6</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>5→6</td>
<td>0</td>
<td>50</td>
</tr>
</tbody>
</table>
ILP for Partitioning - Communication Modeling

- Introduce new binary variables
  \[ g_{i,j} = 1, \text{ iff the communication from node } i \text{ to node } j \text{ uses the bus} \]

- Define \( g_{i,j} \) in terms of the primary variables \( x_{i,k} \)

\[
x_{i,1} \cdot x_{j,2} + x_{i,2} \cdot x_{j,1} = g_{i,j} \quad \forall \text{ edges } (v_i, v_j)
\]

- Linearization

\[
\begin{align*}
x_{i,1} + x_{j,2} - 1 & \leq g_{i,j} \\
x_{i,2} + x_{j,1} - 1 & \leq g_{i,j}
\end{align*}
\quad \forall \text{ edges } (v_i, v_j)
\]

- New cost function

\[
\min \sum_{i=1}^{n} \left( c_{i,1} \cdot x_{i,1} + c_{i,2} \cdot x_{i,2} \right) + \sum_{\text{edges}(v_i, v_j)} \text{com}_{i,j} \cdot g_{i,j}
\]
Example: Hw/Sw Bi-Partitioning (8)

```
/* communication costs */
/***********************/
x12 + x21 - 1 <= g12;
x11 + x22 - 1 <= g12;
x12 + x31 - 1 <= g13;
x11 + x32 - 1 <= g13;
x22 + x41 - 1 <= g24;
x21 + x42 - 1 <= g24;
x22 + x51 - 1 <= g25;
x21 + x52 - 1 <= g25;
x32 + x51 - 1 <= g35;
x31 + x52 - 1 <= g35;
x42 + x61 - 1 <= g46;
x41 + x62 - 1 <= g46;
x52 + x61 - 1 <= g56;
x51 + x62 - 1 <= g56;
/**********************/
/* variables in {0,1} */
**********************/
g12 <= 1;
g13 <= 1;
g24 <= 1;
g25 <= 1;
g35 <= 1;
g46 <= 1;
g56 <= 1;
/**********************/
/* integer variables */
**********************/
int g12;
in g13;
int g24;
in g25;
int g35;
in g46;
in g56;
/**********************/
/* objective function */
/***********************/
/* c1s = 80  c1h = 320 */
/* c2s = 240  c2h = 170 */
/* c3s = 710  c3h = 120 */
/* c4s = 130  c4h = 20  */
/* c5s = 100  c5h = 400 */
/* c6s = 80   c6h = 260 */
g12 = 20 */
g13 = 250 */
g24 = 200 */
g25 = 10 */
g35 = 100 */
g46 = 5 */
g56 = 50 */
/* minimize */
min: 80 x11 + 320 x12 + 240 x21 + 170 x22 + 710 x31 + 120 x32 + 130 x41 + 20 x42 + 100 x51 + 400 x52 + 80 x61 + 260 x62 + 20 g12 + 250 g13 + 200 g24 + 10 g25 + 100 g35 + 5 g46 + 50 g56;
```
**Example: Hw/Sw Bi-Partitioning (9)**

allocation & binding

<table>
<thead>
<tr>
<th>task</th>
<th>SW cost</th>
<th>HW cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>80</td>
<td>320</td>
</tr>
<tr>
<td>2</td>
<td>240</td>
<td>170</td>
</tr>
<tr>
<td>3</td>
<td>710</td>
<td>120</td>
</tr>
<tr>
<td>4</td>
<td>130</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>400</td>
</tr>
<tr>
<td>6</td>
<td>80</td>
<td>260</td>
</tr>
</tbody>
</table>

**cost table**

- n=6
- m=2
- constraint: HW cost <= 300
- total cost = 1100
Overview

• Integer Linear Programs (ILP) basics

• ILP for the partitioning problem (system synthesis)

• ILP for architecture synthesis (hardware synthesis)
Example: DFG (1)

- $G_S(V_S, E_S)$

All operations take one time step.

Resource constraints:
- 2 MUL
- 2 ALU ($+, -, <$)

→ Find schedule with minimal latency.

$V_S = \{v_0, v_1, ..., v_{12}\}$

Alternative problem: minimize resources under latency constraint.
ILP for Architecture Synthesis (1)

- **Binary variables**
  - \( x_{i,l} = 1 \), iff operation \( v_i \) starts in time step \( l, \ i = 1, \ldots, n \quad l = 1, \ldots, L \)
  - \( \rightarrow \) we need to define a latency bound

- **Uniqueness constraints**
  - each operation starts in exactly one time step

  \[
  \sum_{l} x_{i,l} = 1, \ \forall \ i = 1, \ldots, n
  \]

- **Resource constraints**
  - in each time step, use no more resources than available

  \[
  \sum_{\beta(v_i)=v_{MUL}} x_{i,l} \leq \alpha(v_{MUL}), \quad \sum_{\beta(v_i)=v_{ALU}} x_{i,l} \leq \alpha(v_{ALU}), \ \forall \ l = 1, \ldots, L
  \]
**ILP for Architecture Synthesis (2)**

- **Sequencing constraints**
  - an operation cannot start earlier than the finishing time of its predecessors

\[
\sum_l l \cdot x_{j,l} \geq w(v_i, \beta(v_i)) + \sum_l l \cdot x_{i,l}, \quad \forall \text{ edges } (v_i, v_j)
\]

- **Cost function**
  - minimize start time of last operation

\[
\min \sum_l l \cdot x_{n,l}
\]
Example: DFG (2)

```c
/*********************/
/* integer variables */
/*********************/
int x1_1;
int x1_2;
int x1_3;
int x1_4;
int x1_5;
int x1_6;
int x1_7;
int x2_1;
int x2_2;
int x2_3;
int x2_4;
int x2_5;
int x2_6;
int x2_7;
int x3_1;
int x3_2;
int x3_3;
int x3_4;
int x3_5;
int x3_6;
int x3_7;
int x4_1;
int x4_2;
int x4_3;
int x4_4;
int x4_5;
int x4_6;
int x4_7;
int x5_1;
int x5_2;
int x5_3;
int x5_4;
int x5_5;
int x5_6;
int x5_7;
int x6_1;
int x6_2;
int x6_3;
int x6_4;
int x6_5;
int x6_6;
int x6_7;
int x7_1;
int x7_2;
int x7_3;
int x7_4;
int x7_5;
int x7_6;
int x7_7;
int x8_1;
int x8_2;
int x8_3;
int x8_4;
int x8_5;
int x8_6;
int x8_7;
int x9_1;
int x9_2;
int x9_3;
int x9_4;
int x9_5;
int x9_6;
int x9_7;
int x10_1;
int x10_2;
int x10_3;
int x10_4;
int x10_5;
int x10_6;
int x10_7;
int x11_1;
int x11_2;
int x11_3;
int x11_4;
int x11_5;
int x11_6;
int x11_7;
int x12_1;
int x12_2;
int x12_3;
int x12_4;
int x12_5;
int x12_6;
int x12_7;
/*********************/
/* variables in {0,1} */
/*********************/
x1_1 <= 1;
x1_2 <= 1;
x1_3 <= 1;
x1_4 <= 1;
x1_5 <= 1;
x1_6 <= 1;
x1_7 <= 1;
x2_1 <= 1;
x2_2 <= 1;
x2_3 <= 1;
x2_4 <= 1;
x2_5 <= 1;
x2_6 <= 1;
x2_7 <= 1;
x3_1 <= 1;
x3_2 <= 1;
x3_3 <= 1;
x3_4 <= 1;
x3_5 <= 1;
x3_6 <= 1;
x3_7 <= 1;
x4_1 <= 1;
x4_2 <= 1;
x4_3 <= 1;
x4_4 <= 1;
x4_5 <= 1;
x4_6 <= 1;
x4_7 <= 1;
x5_1 <= 1;
x5_2 <= 1;
x5_3 <= 1;
x5_4 <= 1;
x5_5 <= 1;
x5_6 <= 1;
x5_7 <= 1;
x6_1 <= 1;
x6_2 <= 1;
x6_3 <= 1;
x6_4 <= 1;
x6_5 <= 1;
x6_6 <= 1;
x6_7 <= 1;
x7_1 <= 1;
x7_2 <= 1;
x7_3 <= 1;
x7_4 <= 1;
x7_5 <= 1;
x7_6 <= 1;
x7_7 <= 1;
x8_1 <= 1;
x8_2 <= 1;
x8_3 <= 1;
x8_4 <= 1;
x8_5 <= 1;
x8_6 <= 1;
x8_7 <= 1;
x9_1 <= 1;
x9_2 <= 1;
x9_3 <= 1;
x9_4 <= 1;
x9_5 <= 1;
x9_6 <= 1;
x9_7 <= 1;
x10_1 <= 1;
x10_2 <= 1;
x10_3 <= 1;
x10_4 <= 1;
x10_5 <= 1;
x10_6 <= 1;
x10_7 <= 1;
x11_1 <= 1;
x11_2 <= 1;
x11_3 <= 1;
x11_4 <= 1;
x11_5 <= 1;
x11_6 <= 1;
x11_7 <= 1;
x12_1 <= 1;
x12_2 <= 1;
x12_3 <= 1;
x12_4 <= 1;
x12_5 <= 1;
x12_6 <= 1;
x12_7 <= 1;
latency bound = 7
```
Example: DFG (3)

**************************
/* uniqueness constraints */
**************************
x1_1 + x1_2 + x1_3 + x1_4 + x1_5 + x1_6 + x1_7 = 1;
x2_1 + x2_2 + x2_3 + x2_4 + x2_5 + x2_6 + x2_7 = 1;
x3_1 + x3_2 + x3_3 + x3_4 + x3_5 + x3_6 + x3_7 = 1;
x4_1 + x4_2 + x4_3 + x4_4 + x4_5 + x4_6 + x4_7 = 1;
x5_1 + x5_2 + x5_3 + x5_4 + x5_5 + x5_6 + x5_7 = 1;
x6_1 + x6_2 + x6_3 + x6_4 + x6_5 + x6_6 + x6_7 = 1;
x7_1 + x7_2 + x7_3 + x7_4 + x7_5 + x7_6 + x7_7 = 1;
x8_1 + x8_2 + x8_3 + x8_4 + x8_5 + x8_6 + x8_7 = 1;
x9_1 + x9_2 + x9_3 + x9_4 + x9_5 + x9_6 + x9_7 = 1;
x10_1 + x10_2 + x10_3 + x10_4 + x10_5 + x10_6 + x10_7 = 1;
x11_1 + x11_2 + x11_3 + x11_4 + x11_5 + x11_6 + x11_7 = 1;
x12_1 + x12_2 + x12_3 + x12_4 + x12_5 + x12_6 + x12_7 = 1;

**********************/
/* resource constraints */
***********************/
x1_1 + x2_1 + x3_1 + x4_1 + x6_1 + x7_1 <= 2;
x5_1 + x8_1 + x9_1 + x10_1 + x11_1 <= 2;
x1_2 + x2_2 + x3_2 + x4_2 + x6_2 + x7_2 <= 2;
x5_2 + x8_2 + x9_2 + x10_2 + x11_2 <= 2;
x1_3 + x2_3 + x3_3 + x4_3 + x6_3 + x7_3 <= 2;
x5_3 + x8_3 + x9_3 + x10_3 + x11_3 <= 2;
x1_4 + x2_4 + x3_4 + x4_4 + x6_4 + x7_4 <= 2;
x5_4 + x8_4 + x9_4 + x10_4 + x11_4 <= 2;
x1_5 + x2_5 + x3_5 + x4_5 + x6_5 + x7_5 <= 2;
x5_5 + x8_5 + x9_5 + x10_5 + x11_5 <= 2;
x1_6 + x2_6 + x3_6 + x4_6 + x6_6 + x7_6 <= 2;
x5_6 + x8_6 + x9_6 + x10_6 + x11_6 <= 2;
x1_7 + x2_7 + x3_7 + x4_7 + x6_7 + x7_7 <= 2;
x5_7 + x8_7 + x9_7 + x10_7 + x11_7 <= 2;

/**********************/
/* objective function */
***********************/
min: 1 x12_1 + 2 x12_2 + 3 x12_3 + 4 x12_4 + 5 x12_5 + 6 x12_6 + 7 x12_7;
Example: DFG (4)
Example: DFG - Optimal Schedule

latency = 4