

*The OVAKO Application  
Constraint Model and Implementation*

Mats Carlsson, SICS

TACIT, ESPRIT Project 23365

Review meeting, Mons, February 19, 1998

**TACIT**

**Trial Application using Constraint programming in Industrial manufacturing**

# *Suites*

- A *suite* is defined as part of a batch assigned to with part of a customer order.
- There are two suite types, *square* and *round*, corresponding to the two flows.
- Ingots belonging to two suites of the same type may not be interleaved.
- Ingots belonging to two suites of different type may be interleaved.
- Ingots are numbered consecutively within a suite and a batch.

## *Constants*

CDUR Load duration of an ingot.

DDUR Unload duration of an ingot.

CYCLE( $j$ ) The minimal time between successive unloads in suite  $j$ .

SETUP( $p, q$ ) Setup time for suite  $q$  preceded by suite  $p$ . This happens when roller stand 3 needs reconfiguring.

# *Parameters*

NJ Number of suites.

NB Number of batches.

REL( $b$ ) Release time for batch  $b$ .

SUITES( $b$ ) The set of suites of batch  $b$ .

TYPE( $j$ ) The type of suite  $j$ .

Is( $j$ ) The set of ingots of suite  $j$ .

HDUR( $j, f$ ) The heating duration for suite  $j$  and furnace  $f$ .

SDUR( $j$ ) The soaking duration for suite  $j$ .

BIs( $b$ ) Shorthand for  $\bigcup_{j \in \text{SUITES}(b)} \text{Is}(j)$ .

## *Variables*

$F(b) \in 1..12$  Furnace # of batch  $b$ .

$T_c(i) \in 0..\infty$  Load time for ingot  $i$ .

$T_d(i) \in 0..\infty$  Unload time for ingot  $i$ .

## *Load time constraint*

$\forall b \in 1..NB :$

$$\forall k \in 0..n : T_c(i_k) = \text{REL}(b) + k \times \text{CDUR}$$

where

$$\text{BIs}(b) = i_0..i_n$$

## *Unload order constraint*

Break symmetries by imposing an unload order within each suite.

$$\forall j \in 1..NJ$$

$$T_d(i_0) + \text{CYCLE}(j) \leq \dots \leq T_d(i_n)$$

where

$$\text{Is}(j) = i_0..i_n$$

## *Heating and soaking time constraint*

$\forall b \in 1..NB, j \in \text{SUITES}(b)$

$$\text{REL}(b) + \text{HDUR}(j, F(b)) + \text{SDUR}(j) \leq T_d(i_0)$$

where

$$\text{Is}(j) = i_0..i_n$$

## *Exclusive use of furnaces*

$\forall b, b' \in 1..NB \mid b < b' :$

$$F(b) \neq F(b') \vee$$

$$\forall j \in \text{SUITES}(b) : T_d(i_n) + \text{DDUR} \leq \text{REL}(b') \vee$$

$$\forall j \in \text{SUITES}(b') : T_d(i_n) + \text{DDUR} \leq \text{REL}(b)$$

where

$$\text{Is}(j) = i_0..i_n$$

## *Non-overlapping unloading of suites*

$\forall i, j \in 1..NJ \mid i < j :$

$$\text{TYPE}(i) \neq \text{TYPE}(j) \vee$$

$$T_d(i_n) + \text{CYCLE}(i) + \text{SETUP}(i, j) \leq T_d(j_0) \vee$$

$$T_d(j_{n'}) + \text{CYCLE}(j) + \text{SETUP}(j, i) \leq T_d(i_0)$$

where

$$\text{Is}(i) = i_0..i_n, \text{Is}(j) = j_0..j_{n'}$$

## *Load-unload constraints*

$\forall b, b' \in 1..NB \mid b \neq b' :$

$$C(F(b), F(b')) \vee$$

$\forall j \in \text{SUITES}(b') :$

$$\text{REL}(b) + |\text{BIs}(b)| \times \text{CDUR} \leq T_d(i_0) \vee$$

$$T_d(i_n) + \text{DDUR} \leq \text{REL}(b)$$

where

$$\text{Is}(j) = i_0..i_n$$

The compatibility relation  $C$  is defined on the next slide.

## *Load-unload constraint contd.*

Simultaneous loading and unloading is allowed if the furnaces are not too close.

$$C(x, y) \stackrel{\Delta}{=} \left( \begin{array}{l} x \in 1..12 \wedge y \in 1..12 \wedge \\ (x = 1 \wedge y \geq 4) \vee \\ (x = 2 \wedge y \geq 4) \vee \\ (x = 3 \wedge y \geq 5) \vee \\ (x = 4 \wedge y \geq 5) \vee \\ (x = 5 \wedge y \neq 5 \wedge y \neq 6) \vee \\ (x = 6 \wedge y \neq 5 \wedge y \neq 6) \vee \\ (x = 7 \wedge y \neq 7 \wedge y \neq 8) \vee \\ (x = 8 \wedge y \neq 7 \wedge y \neq 8) \vee \\ (x = 9 \wedge y \neq 9 \wedge y \neq 10) \vee \\ (x = 10 \wedge y \neq 9 \wedge y \neq 10) \vee \\ (x = 11 \wedge y \leq 10) \vee \\ (x = 12 \wedge y \leq 10) \end{array} \right)$$

## *Cost function*

Furnaces tend to be bottlenecks, so minimize their total busy time.

$$\text{Cost} = \sum_{b \in 1..NB} (\max\{T_d(i) \mid i \in \text{BIs}(b)\} - \text{REL}(b))$$

Alternatives: minimize reconfigurations, minimize energy consumption, ...

## *Implementation: Programming environment*

SICStus Prolog 3 and its finite domain constraint solver, and interface to Tcl/Tk 8.0. Reference:

M. Carlsson, G. Ottosson, B. Carlson, “An Open-Ended Finite Domain Constraint Solver”, Proc. Programming Languages: Implementations, Logics, and Programs, LNCS 1292, Springer-Verlag, 1997.

# Implementation: A Greedy Algorithm

```
Greedy( $B$ )                                     {  $B$  is the set of batches }
{
     $S \leftarrow \emptyset$                        {  $S$  is the schedule being built }
L1:   if  $B = \emptyset$  return  $S$ 
      select  $b \in B$  such that  $\text{REL}(b)$  is minimal
       $B \leftarrow B \setminus \{b\}$ 
       $S \leftarrow S' \in \{\text{Augment}(S, b, f) \mid f \in 1..12\}$  such that  $\min(\text{Cost}(S'))$  is minimal
      goto L1
}
```

```
Augment( $S, b, f$ )                               { Augment  $S$  assuming batch  $b$  in furnace  $f$  }
{
    assign  $b$  to the furnace  $f$ 
    ( $J_r, J_s$ )  $\leftarrow$  the round/square suites of  $b$ 
L2:   if  $J_r = \emptyset$  goto L2                 { the round flow is a bottleneck }
      select  $j \in J_r$  such that the earliest unload time is minimal
       $J_r \leftarrow J_r \setminus \{j\}$ 
      schedule  $j$  as early as possible into  $S$ 
      goto L2
L3:   if  $J_s = \emptyset$  return  $S$              { exploit parallelism with the round flow }
      select  $j \in J_s$  such that the earliest unload time is minimal
       $J_s \leftarrow J_s \setminus \{j\}$ 
      schedule  $j$  as early as possible into  $S$ 
      goto L3                                   {  $f$  is completely unloaded now }
}
```

# *A Refined Implementation*

1. Include previous SICS work on order assignment in the OVAKO domain.
2. Use variable load times (but load ASAP).
3. Use the greedy algorithm to produce a furnace assignment, a unload order per furnace, and two global unload orders, for square and round suites.
4. Use a local search procedure to improve the choices made by the greedy algorithm.
5. Use published techniques, if applicable.