A hybrid metaheuristic for production planning

João Pedro PEDROSO
Universidade do Porto, Portugal
jpp@ncc.up.pt

Makoto OHNISHI
Fujitsu Research Institute, Japan
ohnishi@fri.fujitsu.com

Mikio KUBO
Tokyo University of Marine Science and Technology, Japan
kubo@e.kaiyodai.ac.jp

MIC, Vienna, August 2005
Introduction

This work deals with two problems arising in production planning:

- **lot sizing**
- **scheduling**

- usually these problems are treated separately
- for both problems: exact solution can be rather hard
- appropriate solvers are different:
  - lot sizing $\rightarrow$ mixed integer programming (MIP)
  - scheduling $\rightarrow$ constraint programming
- metaheuristics: provide a unified framework
- this work: focus on the *integration*
Motivation

● Practical problem:
  – large industry
  – stable demand
  – production site where raw materials are transformed into end products.

● Currently:
  – scheduling operations come from customer orders
  – scheduling based on feasibility: no notion of cost involved
  – demand is stable → why not think about lot sizes?

● Aim:
  – formalise the problem
  – lot sizing + scheduling → scheduling operations derived from good/optimal lot sizes
  – implement a prototype
  – check feasibility of the approach with nearly-real data

● Planning:
  – Short term (scheduling): monthly basis
  – Medium term (lot sizing): yearly basis
Background

Previous work in this area: LISCOS European project

- Exact approaches
- MIP for lot sizing
- Constraint programming for scheduling
- Both are commercial solvers
- Cost → not appropriate for prototyping

→ metaheuristics
Lot sizing

Considering all the orders, for the whole of the planning horizon, decide:

- quantity of each lot to be produced
- when to produce each lot
- (not concerned with order of production in the machines)
Scheduling

For each operation of a given period of the lot sizing problem:

- assign it to a machine
- assign it an order in the operations of that machine
- detail: machines can operate in several modes:
  - full capacity $\rightarrow$ higher cost
  - reduced capacity $\rightarrow$ lower cost
Time horizons

- are different for lot sizing and for scheduling
- horizon for scheduling $\leftrightarrow$ one period of lot sizing model
- usually: scheduling only for the first period of lot sizing
Main solution procedure

Start → Solve lot sizing prob → Prepare scheduling problem

Add constraint cutting current solution

Feasible? N → Start Y → Stop

Solve scheduling problem
Lot sizing model

- Costs:
  - setup (fixed) costs
  - variable production costs
  - inventory
  - backlog

- Decision variables:
  - manufacture or not of a product in each period: setup, binary variable $y_{pmt}$
    * $y_{pmt} = 1$ if product $p$ is manufactured in machine $m$ during period $t$
    * $y_{pmt} = 0$ otherwise
  - amount produced: continuous variable $x_{pmt}$
    * corresponding to $y_{pmt}$.
    * $x_{pmt} > 0 \Rightarrow y_{pmt} = 1$
  - inventory $h_{pt}$ and backlog $g_{pt}$
Objective

setup costs: \( F = \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} \sum_{t \in \mathcal{T}} f_{pmt} y_{pmt} \)
- \( f_{pmt} \) is the cost of setting up machine \( m \) on period \( t \) for producing \( p \)

variable costs: \( V = \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} \sum_{t \in \mathcal{T}} v_{pmt} x_{pmt} \)
- \( v_{pmt} \) is the variable cost of production of \( p \) on machine \( m \), period \( t \)

inventory costs: \( I = \sum_{p \in \mathcal{P}} \sum_{t \in \mathcal{T}} i_{pt} h_{pt} \)
- \( h_{pt} \) is the amount of product \( p \) that is kept in inventory at the end of period \( t \)
- \( i_{pt} \) is the unit inventory cost for product \( p \) on period \( t \)

backlog costs: \( B = \sum_{p \in \mathcal{P}} \sum_{t \in \mathcal{T}} b_{pt} g_{pt} \)
- \( g_{pt} \) is the amount of product \( p \) that failed to meet demand at the end of period \( t \)
- \( b_{pt} \) is the unit backlog cost for product \( p \) on period \( t \).

objective: minimise \( z = F + V + I + B \)
Constraints:

flow conservation:

\[ h_{p,t-1} - g_{p,t-1} + \sum_{m \in \mathcal{M}^p} x_{pmt} = D_{pt} + h_{pt} - g_{pt} \quad \forall \ p \in \mathcal{P}, \ \forall \ t \in \mathcal{T}. \]

\( h_{p0}, \ h_{pT} \): initial and final inventory
\( g_{p0}, \ g_{pT} \): initial and final backlog

time availability on each period:

\[ \sum_{p \in \mathcal{P} : m \in \mathcal{M}^p} \left( \frac{x_{pmt}}{\gamma_{pm}} + \tau_{pmt} y_{pmt} \right) \leq A_{mt} \quad \forall \ m \in \mathcal{M}, \ \forall \ t \in \mathcal{T}. \]

\( \gamma_{pm} \) is the total capacity of production of product \( p \) on machine \( m \) per time unit
\( \tau_{pmt} \) is the setup time required if there is production of \( p \) on machine \( m \) during period \( t \)
\( A_{mt} \) is the number of time units available for production on machine \( m \) during period \( t \).

setup constraints:

\[ x_{pmt} \leq \gamma_{pm} A_{mt} y_{pmt} \]
minimise \[ z = F + V + I + B \]

subject to:

\[ F = \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} \sum_{t \in \mathcal{T}} f_{pmt} \ y_{pmt} \]
\[ V = \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} \sum_{t \in \mathcal{T}} v_{pmt} \ x_{pmt} \]
\[ I = \sum_{p \in \mathcal{P}} \sum_{t \in \mathcal{T}} i_{pt} \ h_{pt} \]
\[ B = \sum_{p \in \mathcal{P}} \sum_{t \in \mathcal{T}} b_{pt} \ g_{pt} \]

\[ h_{p,t-1} - g_{p,t-1} + \sum_{m \in \mathcal{M}^p} x_{pmt} = D_{pt} + h_{pt} - g_{pt}, \quad \forall \ p \in \mathcal{P}, \forall \ t \in \mathcal{T} \]

\[ \sum_{p \in \mathcal{P}: m \in \mathcal{M}^p} \left( \frac{x_{pmt}}{\gamma_{pm}} + \tau_{pmt} \ y_{pmt} \right) \leq A_{mt}, \quad \forall \ m \in \mathcal{M}, \forall \ t \in \mathcal{T} \]

\[ x_{pmt} \leq \gamma_{pm} A_{mt} y_{pmt}, \quad \forall \ p \in \mathcal{P}, \forall \ m \in \mathcal{M}^p, \forall \ t \in \mathcal{T} \]

\( F, V, I, B \in \mathbb{R}^+ \)
\( h_{pt}, g_{pt} \in \mathbb{R}^+, \quad \forall \ p \in \mathcal{P}, \forall \ t \in \mathcal{T} \)
\( x_{pmt} \in \mathbb{R}^+, \ y_{pmt} \in \{0, 1\}, \quad \forall \ p \in \mathcal{P}, \forall \ m \in \mathcal{M}, \forall \ t \in \mathcal{T} \)
Construction: relax-and-fix-one-product

- construction of a solution: based on partial relaxations of the initial problem
- variant of the classic relax-and-fix heuristic
Relax-and-fix

- each period is treated independently
- relax all the variables except those of period 1:
  - keep $y_{pmt}$ integer
  - relax integrity for all other $y_{pmt}$
- solve this MIP, determining heuristic values for $\bar{y}_{pmt}$
Relax-and-fix

- each period is treated independently
- relax all the variables except those of period 1:
  - keep $y_{pm1}$ integer
  - relax integrity for all other $y_{pmt}$
- solve this MIP, determining heuristic values for $\bar{y}_{pm1}$
- move to the second period:
  - variables of the first period are fixed at $y_{pm1} = \bar{y}_{pm1}$
  - variables $y_{pm2}$ are integer
  - and all the other $y_{pmt}$ relaxed
- this determines the heuristic value for $y_{pm2}$
Relax-and-fix

- each period is treated independently
- relax all the variables except those of period 1:
  - keep $y_{pm1}$ integer
  - relax integrity for all other $y_{pmt}$
- solve this MIP, determining heuristic values for $\bar{y}_{pm1}$
- move to the second period:
  - variables of the first period are fixed at $y_{pm1} = \bar{y}_{pm1}$
  - variables $y_{pm2}$ are integer
  - and all the other $y_{pmt}$ relaxed
- this determines the heuristic value for $y_{pm2}$
- these steps are repeated, until all the $y$ variables are fixed
Relax-and-fix

- each period is treated independently
- relax all the variables except those of period 1:
  - keep $y_{pm1}$ integer
  - relax integrity for all other $y_{pmt}$
- solve this MIP, determining heuristic values for $y_{pm1}$
- move to the second period:
  - variables of the first period are fixed at $y_{pm1} = \bar{y}_{pm1}$
  - variables $y_{pm2}$ are integer
  - and all the other $y_{pmt}$ relaxed
- this determines the heuristic value for $y_{pm2}$
- these steps are repeated, until all the $y$ variables are fixed
Relax-and-fix heuristic.

- reported to provide very good solutions for many lot sizing problems
- however, for large instances the exact MIP solution of even a single period can be too time consuming
- we propose a variant where each MIP determines only the variables of one period that concern a single product → relax-and-fix-one-product
Relax-and-fix-one-product variant.

\begin{align*}
\text{relaxall} \ y_{pmt} \ \text{as continuous variables} \\
\text{foreach} \ p \in \mathcal{P} \\
\text{foreach} \ m \in \mathcal{M}^p \\
\text{set} \ y_{pmt} \ \text{as integer} \\
\text{solve MIP} \rightarrow \bar{y}_{pmt}, \forall m \in \mathcal{M}^p \\
\text{foreach} \ m \in \mathcal{M}^p \\
\text{fix} \ y_{pmt} := \bar{y}_{pmt} \\
\text{return} \ \bar{y}
\end{align*}
Relax-and-fix-one-product variant.

**RELAXANDFIXONEPRODUCT()**

1. relax all $y_{pmt}$ as continuous variables
2. for $t = 1$ to $T$
3. foreach $p \in \mathcal{P}$
4. foreach $m \in \mathcal{M}^p$
5. set $y_{pmt}$ as integer
6. solve MIP $\bar{y}_{pmt}, \forall m \in \mathcal{M}^p$
7. foreach $m \in \mathcal{M}^p$
8. fix $y_{pmt} := \bar{y}_{pmt}$
9. return $\bar{y}$
Relax-and-fix-one-product variant.

\[ \text{RelaxAndFixOneProduct() } \]

1. relax all \( y_{pmt} \) as continuous variables
2. \( \text{for } t = 1 \text{ to } T \)
3. \( \text{foreach } p \in \mathcal{P} \)
4. \( \text{foreach } m \in \mathcal{M}^p \)
5. set \( y_{pmt} \) as integer
6. solve MIP \( \bar{y}_{pmt}, \forall m \in \mathcal{M}^p \)
7. \( \text{foreach } m \in \mathcal{M}^p \)
8. fix \( y_{pmt} := \bar{y}_{pmt} \)
9. return \( \bar{y} \)
Relax-and-fix-one-product variant.

\begin{align*}
  &t = 1 \\
  &t = 2 \\
  &\ldots \\
  &t = T
\end{align*}

\textbf{RELAXANDFIXONEPRODUCT()}
\begin{enumerate}
\item relax all $y_{pmt}$ as continuous variables
\item for $t = 1$ to $T$
\item \textbf{foreach} $p \in \mathcal{P}$
\item \textbf{foreach} $m \in \mathcal{M}^p$
\item set $y_{pmt}$ as integer
\item solve MIP $\Rightarrow \bar{y}_{pmt}$, $\forall m \in \mathcal{M}^p$
\item \textbf{foreach} $m \in \mathcal{M}^p$
\item fix $y_{pmt} := \bar{y}_{pmt}$
\item return $\bar{y}$
\end{enumerate}
Relax-and-fix-one-product variant.

\texttt{RelaxAndFixOneProduct()}

1. relax all $y_{pmt}$ as continuous variables
2. \texttt{for} $t = 1 \text{ to } T$
3. \texttt{foreach} $p \in \mathcal{P}$
4. \texttt{foreach} $m \in \mathcal{M}^p$
5. set $y_{pmt}$ as integer
6. solve MIP $\bar{y}_{pmt}, \forall m \in \mathcal{M}^p$
7. \texttt{foreach} $m \in \mathcal{M}^p$
8. fix $y_{pmt} := \bar{y}_{pmt}$
9. return $\bar{y}$
Relax-and-fix-one-product variant.

\begin{align*}
\text{RELAXANDFIXONEPRODUCT()} \\
1. & \text{ relax all } y_{pmt} \text{ as continuous variables} \\
2. & \text{ for } t = 1 \text{ to } T \\
3. & \quad \text{ foreach } p \in \mathcal{P} \\
4. & \quad \quad \text{ foreach } m \in \mathcal{M}^p \\
5. & \quad \quad \text{ set } y_{pmt} \text{ as integer} \\
6. & \quad \quad \text{ solve } \text{MIP} \rightarrow \bar{y}_{pmt}, \forall m \in \mathcal{M}^p \\
7. & \quad \text{ foreach } m \in \mathcal{M}^p \\
8. & \quad \quad \text{ fix } y_{pmt} := \bar{y}_{pmt} \\
9. & \quad \text{ return } \bar{y} \\
\end{align*}
Relax-and-fix-one-product variant.

\begin{tabular}{ll}
  \texttt{t=1} & \texttt{RelaxAndFixOneProduct()} \\
  \texttt{t=2} & (1) relax all $y_{pmt}$ as continuous variables \\
  \ldots & (2) \texttt{for } t = 1 \texttt{ to } T \\
  \texttt{t=T} & (3) \texttt{foreach } p \in \mathcal{P} \\
 & (4) \texttt{foreach } m \in \mathcal{M}^p \\
 & (5) \quad \text{set } y_{pmt} \text{ as integer} \\
 & (6) \quad \text{solve MIP} \rightarrow \bar{y}_{pmt}, \forall m \in \mathcal{M}^p \\
 & (7) \texttt{foreach } m \in \mathcal{M}^p \\
 & (8) \quad \text{fix } y_{pmt} := \bar{y}_{pmt} \\
 & (9) \texttt{return } \bar{y} \\
\end{tabular}

Additional advantage: if repeated, can produce different solutions

\rightarrow \text{repeat it a number of times, retain the best found solution}
Scheduling: solution representation

There are two decisions that have to be taken for specifying a scheduling solution:

- Assigning a machine to each operation
- Establish an order for the operations inside each machine
Assigning a machine to each operation
Operation order for each machine
Solution evaluation (computing makespan and cost)

- Start scheduling operations which do not have free (unscheduled) predecessors
Solution evaluation

- Start scheduling operations which do not have free (unscheduled) predecessors
- Fix their earliest start time and earliest finish time
Solution evaluation

- Start scheduling operations which do not have free (unscheduled) predecessors
- Fix their earliest start time and earliest finish time
- Check operations which can now be scheduled
Solution evaluation

- Start scheduling operations which do not have free (unscheduled) predecessors
- Fix their earliest start time and earliest finish time
- Check operations which can now be scheduled
- Fix their start and finish times
- ...
**Solution evaluation**

- Start scheduling operations which do not have free (unscheduled) predecessors
- Fix their earliest start time and earliest finish time
- Check operations which can now be scheduled
- Fix their start and finish times
- ...  

- **changeover** times/costs
- **transfer** times/costs
- **fixed/variable productions** times/costs
Random solution construction

- Check all operations that can be scheduled

Machine 1

Machine 2

Machine 3
Random solution construction

- Check all operations that can be scheduled
- Randomly select one of them (operation 1)
- Randomly select one of the compatible machines (machine 1)
- Fix this operation
Random solution construction

- (operation 1 is fixed on machine 1)
- Check all operations that can be scheduled (operations 2, 3, 4)
- Randomly select one of them (operation 2)
- Randomly select one of the compatible machines (machine 2)
- Fix this operation
Random solution construction

- (operation 1 is fixed on machine 1)
- (operation 2 is fixed on machine 2)
- Check all operations that can be scheduled (operations 3, 4)
- Randomly select one of them (operation 4)
- Randomly select one of the compatible machines (machine 1)
- Fix this operation
Random solution construction

- (operation 1 is fixed on machine 1)
- (operation 2 is fixed on machine 2)
- (operation 4 is fixed on machine 1)
- Check all operations that can be scheduled (operations 3, 5)
- Randomly select one of them . . .
- Randomly select one of the compatible machines . . .
- . . .
- Until all operations are scheduled
Random solution construction

- Produces a random, but feasible solution (except for violation of maximal makespan)
- Very easy to implement
- Can produce many different solutions
- If repeated many times: might obtain a good solution
Greedy construction

- Check all operations that can be scheduled
- Compute the current makespan when they are assigned to each of the possible machines
Greedy construction

- Check all operations that can be scheduled
- Compute the current makespan when they are assigned to each of the possible machines
- Select the assignment which induces the smallest makespan
- Fix this operation
Greedy construction

- Check all operations that can be scheduled
- Compute the current makespan when they are assigned to each of the possible machines
- Select the assignment which induces the smallest makespan
- Fix this operation
Greedy construction

- Check all operations that can be scheduled
- Compute the *current makespan* when they are assigned to each of the possible machines
- Select the assignment which induces the *smallest* makespan
- Fix this operation
- ... 
- Continue this way until fixing all the operations
Semi-greedy construction

- As in the greedy construction, we check *all the possibilities* for each operation the can be scheduled
- Compute the current makespan for each of these possibilities
Semi-greedy construction

- As in the greedy construction, we check *all the possibilities* for each operation the can be scheduled
- Compute the current makespan for each of these possibilities
- Then, select just the possibilities that satisfy some criterion
- Create a *Restricted Candidate List* (RCL)
- Randomly select an (operation, machine) pair from the RCL
- Fix that operation on that machine
- ... 
- Continue, until fixing all the operations
An algorithm for repeated construction

\textsc{IteratedSemiGreedy}(N, \bar{t})

1. $t^* = \infty$
2. $c^* = \infty$
3. \textbf{for} $n = 1$ \textbf{to} $N$
4. \hspace{1em} $x = \textsc{SemiGreedyConstruct}()$
5. \hspace{1em} $t = \textsc{Makespan}(x)$
6. \hspace{1em} $c = \textsc{Cost}(x)$
7. \hspace{1em} \textbf{if} ($t < \bar{t}$ \textbf{and} $c < c^*$) \textbf{or} ($t < t^*$ \textbf{and} $t^* > \bar{t}$)
8. \hspace{2em} $x^* = x; \ t^* = t; \ c^* = c$
9. \hspace{1em} \textbf{return} $x^*$
Main solution procedure (integration)

Start → Solve lot sizing prob → Prepare scheduling problem

Add constraint cutting current solution

Feasible? → Y → Stop

N → Add constraint cutting current solution
“No good” cuts

Let

- \( y \in \{0, 1\} \) be a partial MIP solution
- \( S = \{ r : y_r = 1 \} \) represent an assignment of tasks to machines

Then, if scheduling cannot find a feasible solution, add cut:

\[
\sum_{r \in S} y_r \leq |S| - 1
\]
Cuts for capacity adjustment

Let

- \( x_{pmt} \) be the production of item \( p \) on period \( t \), machine \( k \)
- \( \bar{x}_{pmt} \) last MIP solution for these variables
- \( I_{mt} \) the heuristic estimate of machine waiting times

If we cannot find a feasible schedule of the tasks on period \( t \), then

- for the set of machines \( M^* \) which did not respect the allowed makespan
- add cut:

\[
\sum_{p \in P_m} x_{pmt} \leq \sum_{p \in P_m} \bar{x}_{pmt} - I_{mt} \quad \forall m \in M^*
\]
Main solution procedure

Start → Solve lot sizing prob → Prepare scheduling problem

Add constraint cutting current solution

Feasible?

Y → Stop

N → Solve scheduling problem → Feasible?
Conclusion

- Motivation: industrial application on production planning
- Lot sizing and scheduling: exact solution difficult for both problems
- Integrated model: even more difficult
- Integration of the models has in itself a heuristic component
- Proposed metaheuristics: there is potential for improvement, but
- The method quickly provides implementable solutions
- Results are sufficient for the current practical requirements