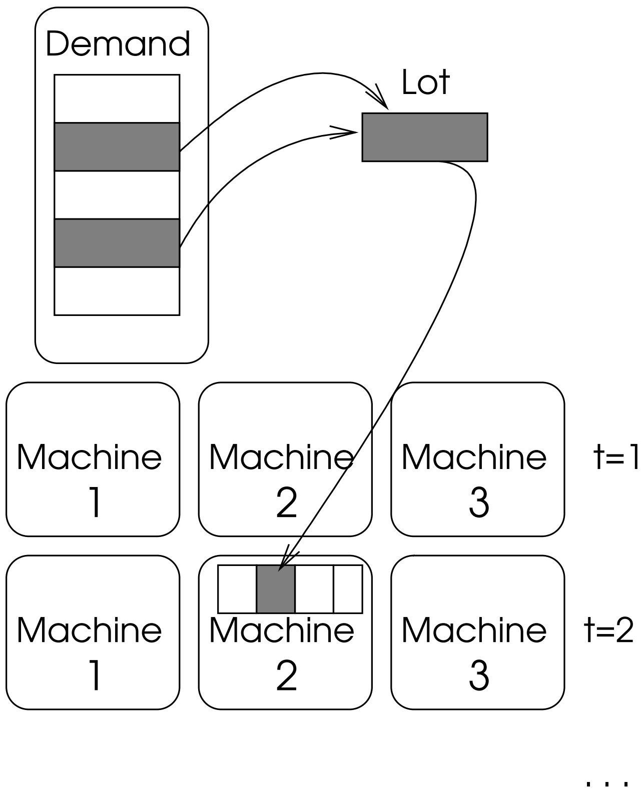
Hybrid tabu search for lot sizing problems

João Pedro Pedroso
Universidade do Porto
jpp@ncc.up.pt
and
Mikio Kubo
Tokyo University of Marine Science and Technology
kubo@e.kaiyodai.ac.jp

HM2005, Barcelona

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)

Lot sizing problems

Ex: for a single product:

period	demand
1	100
2	100
3	100
4	100
5	100
6	100

How should it be produced?

period	production	period	production	period	production
1	100	1	600	1	0
2	100	2	0	2	0
3	100	3	0	3	0
4	100	4	0	4	0
5	100	5	0	5	0
6	100	6	0	6	600

- setup variables
- production variables
- inventory
- backlog

The lot sizing model

big bucket problem: more than one setup allowed per period, as long as machine capacities respected

costs: (values for each of them can vary from period to period)

- setup costs
- variable production costs
- inventory and backlog costs

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}

parameters:

T**:** number of periods, $\mathcal{T} = \{1, \ldots, T\}$

$\mathcal{P}$**:** set of products

$\mathcal{M}$**:** set of machines

$\mathcal{M}^p$**:** subset of machines compatible with the production of p.

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: $\text{minimise}\, z = F + V + I + B$

Constraints:

setup on producing machines:

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

x_{pmt} amount produced
y_{pmt} corresponding setup

time availability on each period:

$$\sum_{p \in \mathcal{P}: m \in \mathcal{M}^p} \quad \frac{x_{pmt}}{\gamma_{pm}} + \tau_{pmt}\, y_{pmt} \quad \leq A_{mt} \qquad \forall\, m \in \mathcal{M},\ \forall\, t \in \mathcal{T}.$$

γ_{pm} is the total capacity of production of product p on machine m per time unit
τ_{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.

flow conservation:

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

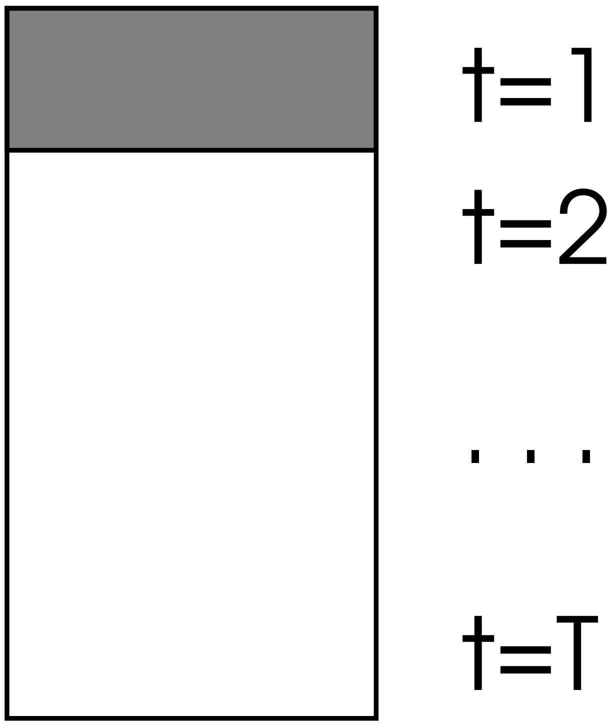
h_{p0}, h_{pT}: initial and final inventory
g_{p0}, g_{pT}: initial and final backlog

$$
\begin{aligned}
\text{minimise} \quad & z = F + V + I + B \\
\text{subject to :} \quad & F = \sum_{p\in\mathcal{P}} \sum_{m\in\mathcal{M}} \sum_{t\in\mathcal{T}} f_{pmt}\; y_{pmt} \\
& V = \sum_{p\in\mathcal{P}} \sum_{m\in\mathcal{M}} \sum_{t\in\mathcal{T}} v_{pmt}\; x_{pmt} \\
& I = \sum_{p\in\mathcal{P}} \sum_{t\in\mathcal{T}} i_{pt}\; h_{pt} \\
& B = \sum_{p\in\mathcal{P}} \sum_{t\in\mathcal{T}} b_{pt}\; g_{pt} \\
& h_{p,t-1} - g_{p,t-1} + \sum_{m\in\mathcal{M}^p} x_{pmt} = D_{pt} + h_{pt} - g_{pt}, \quad \forall\, p \in \mathcal{P},\ \forall\, t \in \mathcal{T} \\
& \sum_{p\in\mathcal{P}:m\in\mathcal{M}^p} \quad \frac{x_{pmt}}{\gamma_{pm}} + \tau_{pmt}\; y_{pmt} \quad \leq A_{mt}, \quad \forall\, m \in \mathcal{M},\ \forall\, t \in \mathcal{T} \\
& x_{pmt} \leq \gamma_{pm}\; A_{mt}\; y_{pmt} \quad \forall\, p \in \mathcal{P},\ \forall\, m \in \mathcal{M}^p,\ \forall\, t \in \mathcal{T} \\
& F, V, I, B \in \mathbb{R}^+ \\
& h_{pt},\ g_{pt} \in \mathbb{R}^+, \quad \forall\, p \in \mathcal{P},\ \forall\, t \in \mathcal{T} \\
& x_{pmt} \in \mathbb{R}^+,\ y_{pmt} \in \{0, 1\}, \quad \forall\, p \in \mathcal{P}, \forall\, m \in \mathcal{M},\ \forall\, t \in \mathcal{T}
\end{aligned}
\tag{1}
$$

Construction: relax-and-fix-one-product

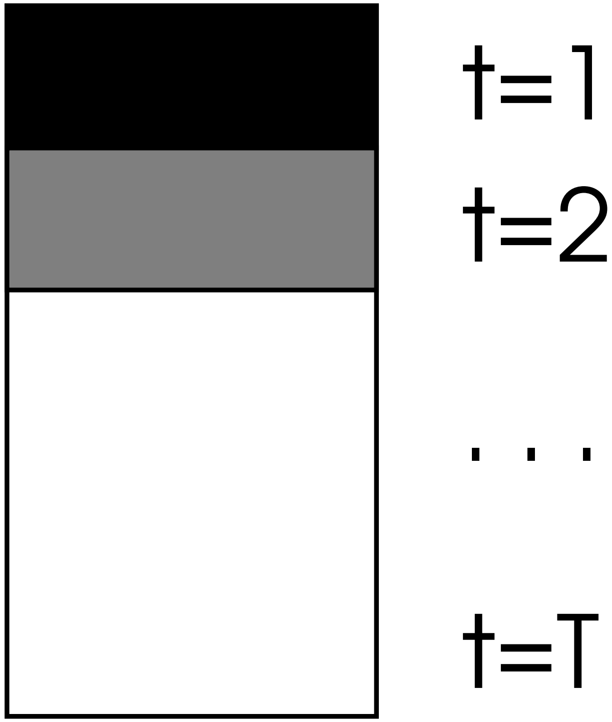
- **construction of a solution:** based on partial relaxations of the initial problem
- variant of the classic relax-and-fix heuristic

Relax-and-fix



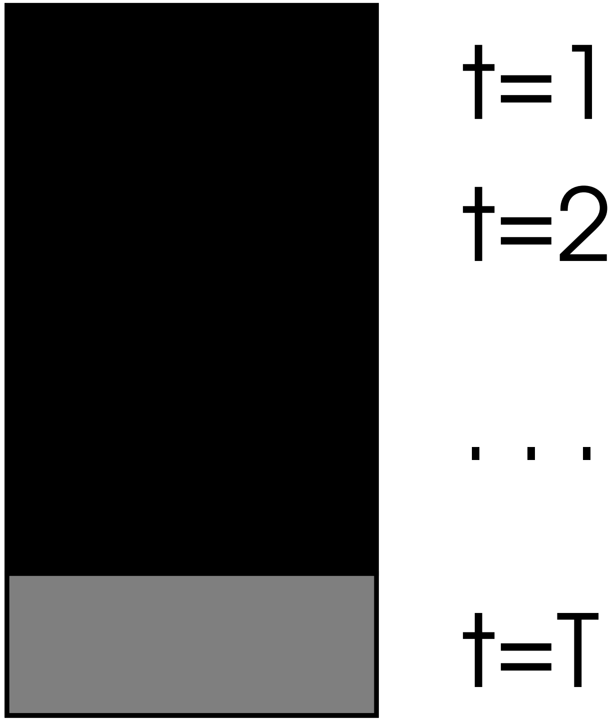
- each period is treated independently
- relax all the variables except those of period 1:
  - keep y_{pm1} integer
  - relax integrity for all other y_{pmt}
- solve this MIP, determining heuristic values for $\bar{y}_{pm1}$

Relax-and-fix



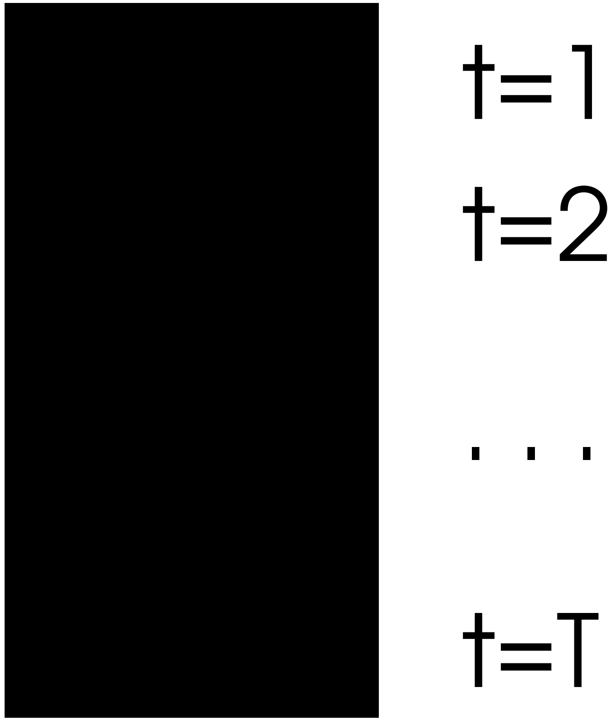
- each period is treated independently
- relax all the variables except those of period 1:
  - keep y_{pm1} integer
  - relax integrity for all other y_{pmt}
- solve this MIP, determining heuristic values for $\bar{y}_{pm1}$
- move to the second period:
  - variables of the first period are fixed at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

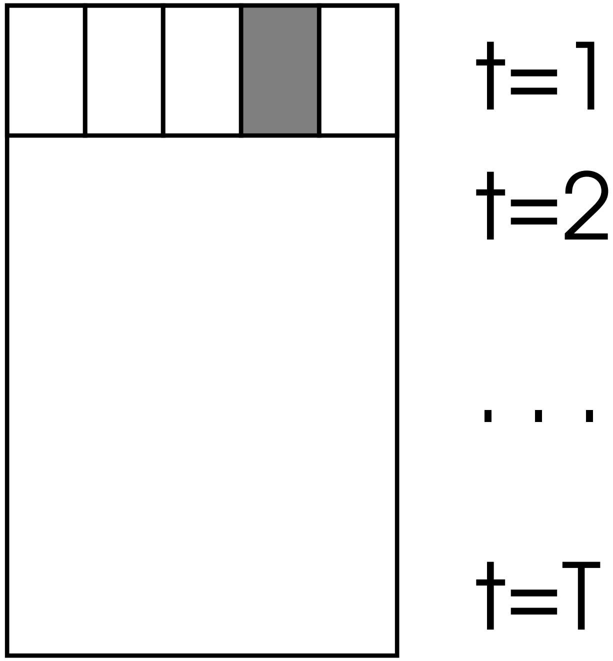


- each period is treated independently
- relax all the variables except those of period 1:
  - keep y_{pm1} integer
  - relax integrity for all other y_{pmt}
- solve this MIP, determining heuristic values for $\bar{y}_{pm1}$
- move to the second period:
  - variables of the first period are fixed at $y_{pm1} = \bar{y}_{pm1}$
  - 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 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 were each MIP determines only the variables of one period *that concern a single product* $\longrightarrow$ **relax-and-fix-one-product**

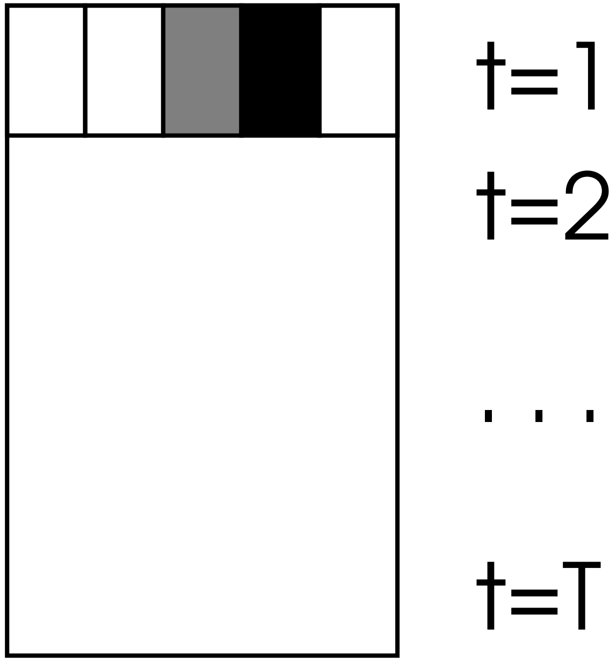
Relax-and-fix-one-product variant.



RELAXANDFIXONEPRODUCT()

relax all y_{pmt} as continuous variables
for $t = 1$ **to** T
    foreach $p \in \mathcal{P}$
        foreach $m \in \mathcal{M}^p$
            set y_{pmt} as integer
        solve MIP$\longrightarrow$ $\bar{y}_{pmt}, \forall m \in \mathcal{M}^p$
        foreach $m \in \mathcal{M}^p$
            fix $y_{pmt} := \bar{y}_{pmt}$
return $\bar{y}$

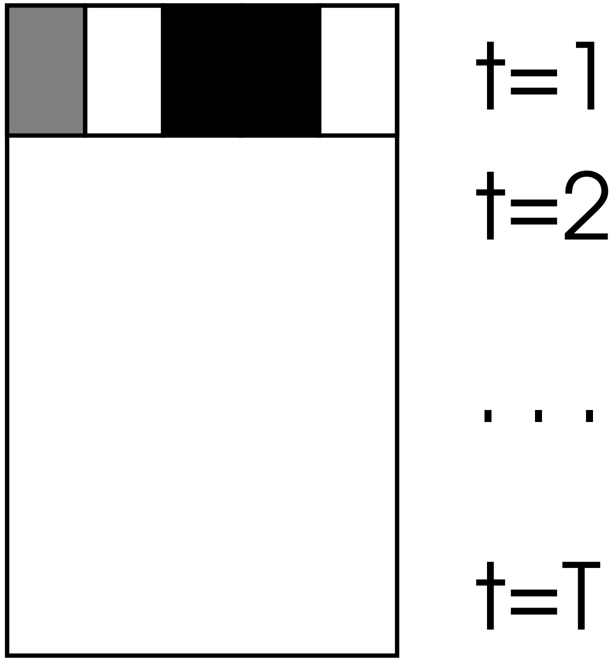
Relax-and-fix-one-product variant.



RELAXANDFIXONEPRODUCT()

relax all y_{pmt} as continuous variables
for $t = 1$ **to** T
    foreach $p \in \mathcal{P}$
        foreach $m \in \mathcal{M}^p$
            set y_{pmt} as integer
        solve MIP$\longrightarrow$ $\bar{y}_{pmt}, \forall m \in \mathcal{M}^p$
        foreach $m \in \mathcal{M}^p$
            fix $y_{pmt} := \bar{y}_{pmt}$
return $\bar{y}$

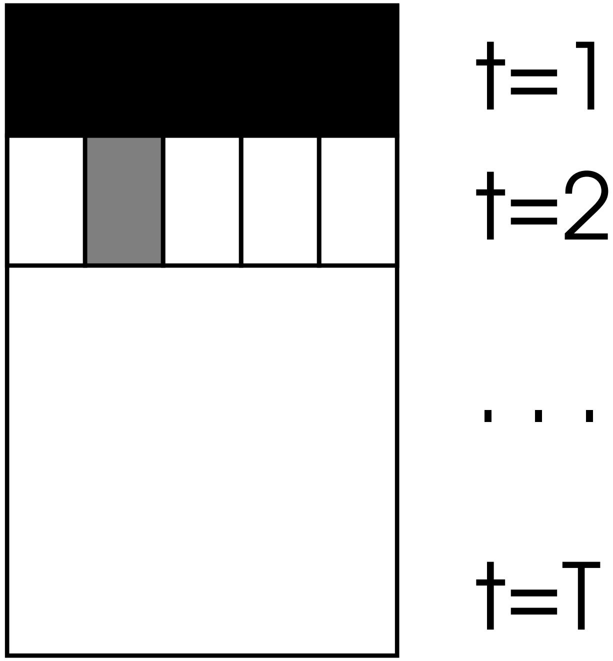
Relax-and-fix-one-product variant.



RELAXANDFIXONEPRODUCT()

relax all y_{pmt} as continuous variables
for $t = 1$ **to** T
    foreach $p \in \mathcal{P}$
        foreach $m \in \mathcal{M}^p$
            set y_{pmt} as integer
        solve MIP$\longrightarrow$ $\bar{y}_{pmt}, \forall m \in \mathcal{M}^p$
        foreach $m \in \mathcal{M}^p$
            fix $y_{pmt} := \bar{y}_{pmt}$
return $\bar{y}$

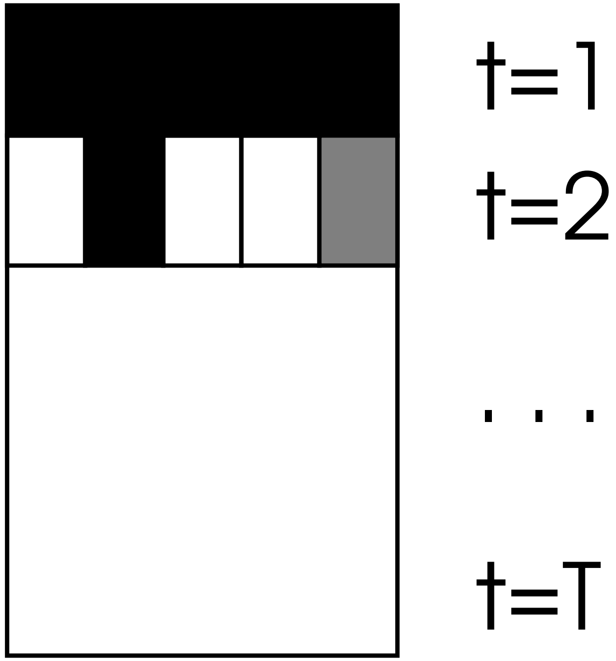
Relax-and-fix-one-product variant.



RELAXANDFIXONEPRODUCT()

relax all y_{pmt} as continuous variables
for $t = 1$ **to** T
    foreach $p \in \mathcal{P}$
        foreach $m \in \mathcal{M}^p$
            set y_{pmt} as integer
        solve MIP$\longrightarrow \bar{y}_{pmt}, \forall m \in \mathcal{M}^p$
        foreach $m \in \mathcal{M}^p$
            fix $y_{pmt} := \bar{y}_{pmt}$
return $\bar{y}$

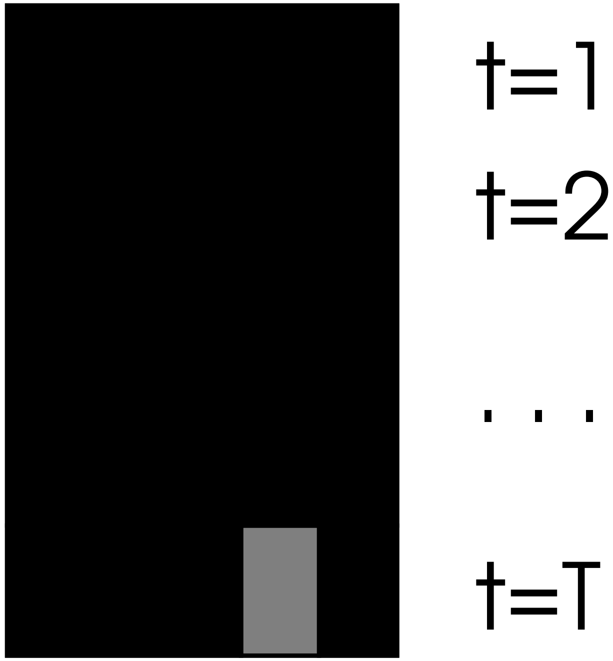
Relax-and-fix-one-product variant.



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$\longrightarrow \bar{y}_{pmt}, \forall m \in \mathcal{M}^p$
(7)   **foreach** $m \in \mathcal{M}^p$
(8)   fix $y_{pmt} := \bar{y}_{pmt}$
(9) **return** $\bar{y}$

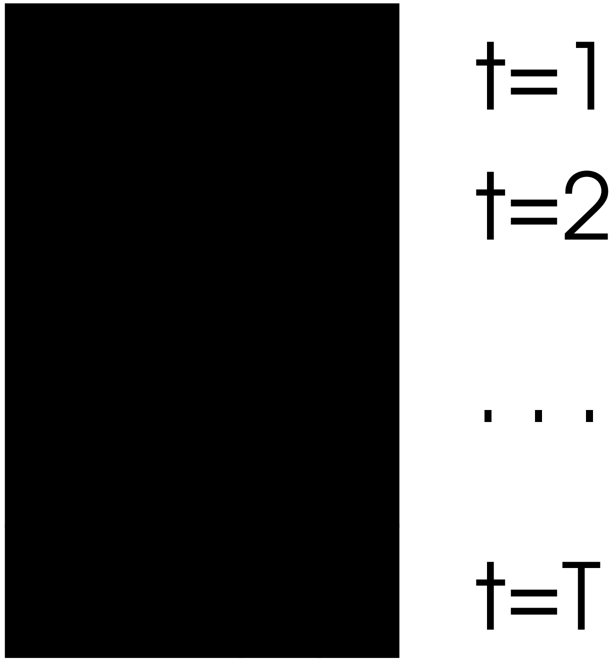
Relax-and-fix-one-product variant.



RelaxAndFixOneProduct()

relax all y_{pmt} as continuous variables
for $t = 1$ **to** T
    foreach $p \in \mathcal{P}$
        foreach $m \in \mathcal{M}^p$
            set y_{pmt} as integer
        solve MIP$\longrightarrow \bar{y}_{pmt}, \forall m \in \mathcal{M}^p$
        foreach $m \in \mathcal{M}^p$
            fix $y_{pmt} := \bar{y}_{pmt}$
return $\bar{y}$

Relax-and-fix-one-product variant.



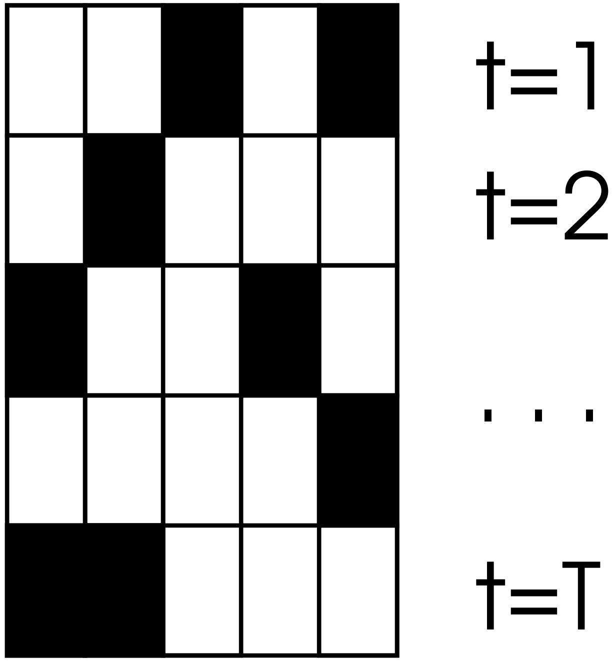
RelaxAndFixOneProduct()

relax all y_{pmt} as continuous variables
for $t = 1$ **to** T
    foreach $p \in \mathcal{P}$
        foreach $m \in \mathcal{M}^p$
            set y_{pmt} as integer
        solve MIP$\longrightarrow \bar{y}_{pmt}, \forall m \in \mathcal{M}^p$
        foreach $m \in \mathcal{M}^p$
            fix $y_{pmt} := \bar{y}_{pmt}$
return $\bar{y}$

Additional advantage: if repeated, can produce different solutions
$\longrightarrow$ repeat it a number of times, retain the best found solution

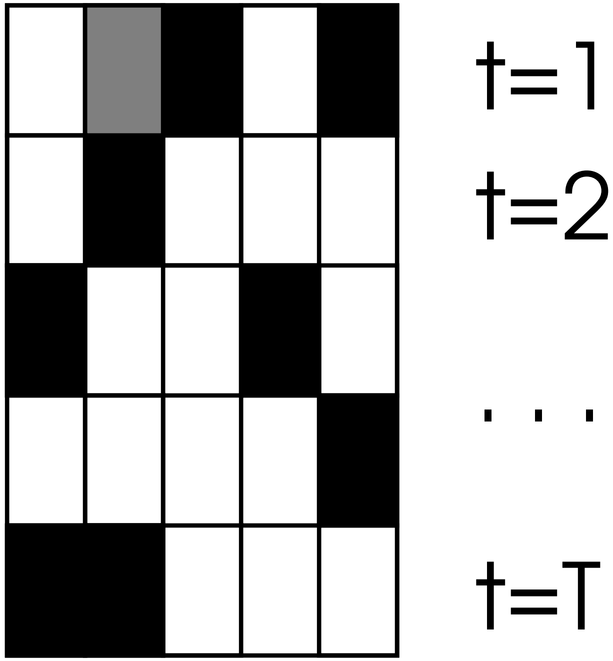
Solution reconstruction

- relax-and-fix-one-product construction mechanism can be used for *completing a solution that has been partially destructed*
- check if incoming $\bar{y}_{pmt}$ variables are initialized or not;
  - if they are initialized, they should be fixed in the MIP at their current value
  - otherwise, they are treated as in the previous algorithm:
    - made integer if they belong to the period and product currently being dealt
    - relaxed otherwise



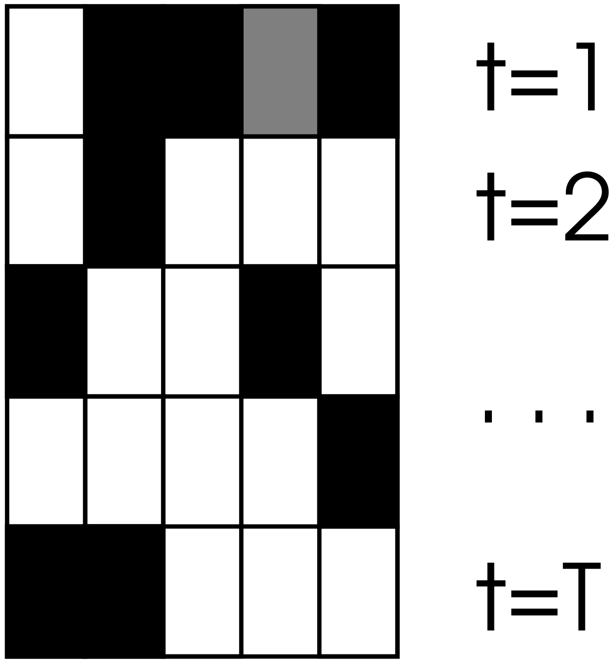
$\textsc{Reconstruct}(\bar{y})$

for $t = 1$ **to** T
foreach $p \in \mathcal{P}$
foreach $m \in \mathcal{M}^p$
if $\bar{y}_{pmt}$ is not initialized
relax y_{pmt}
else
fix $y_{pmt} := \bar{y}_{pmt}$
for $t = 1$ **to** T
foreach $p \in \mathcal{P}$
$\mathcal{U} := \{\}$
foreach $m \in \mathcal{M}^p$
if $\bar{y}_{pmt}$ is not initialized
set y_{pmt} as integer
$\mathcal{U} := \mathcal{U} \cup \{(p, m, t)\}$
solve lot sizing MIP
foreach $(p, m, t) \in \mathcal{U}$
fix $y_{pmt} := \bar{y}_{pmt}$
return $\bar{y}$

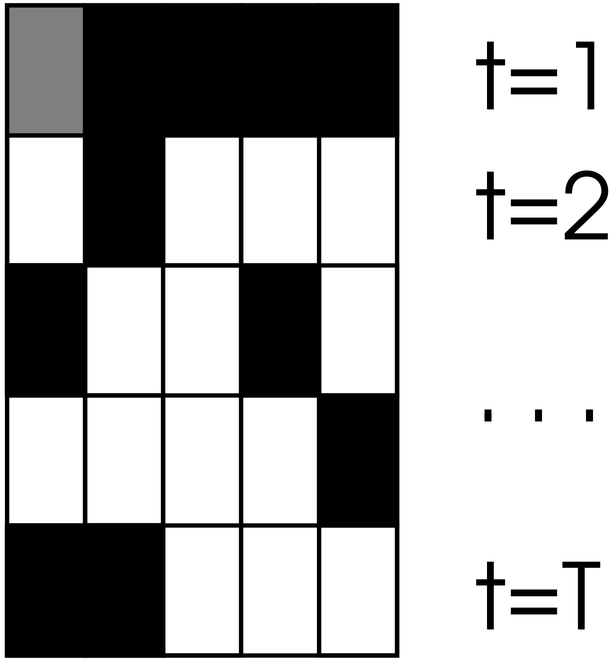


RECONSTRUCT($\bar{y}$)

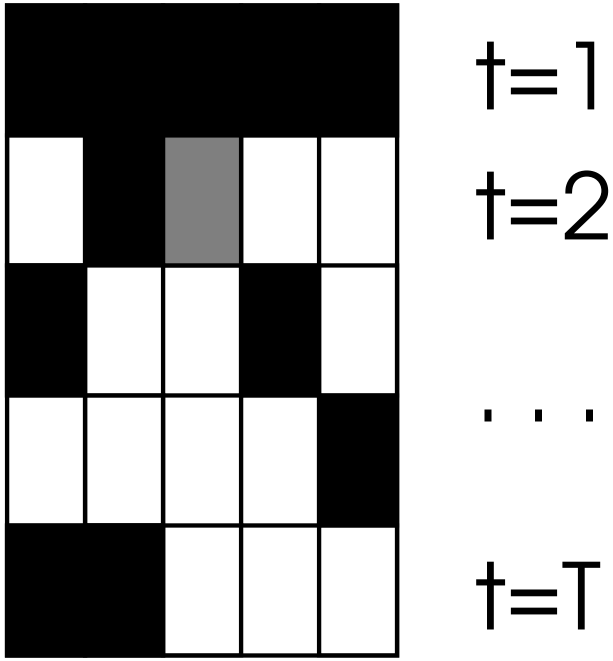
for $t = 1$ **to** T
  foreach $p \in \mathcal{P}$
    foreach $m \in \mathcal{M}^p$
      if $\bar{y}_{pmt}$ is not initialized
        relax y_{pmt}
      else
        fix $y_{pmt} := \bar{y}_{pmt}$
for $t = 1$ **to** T
  foreach $p \in \mathcal{P}$
    $\mathcal{U} := \{\}$
    foreach $m \in \mathcal{M}^p$
      if $\bar{y}_{pmt}$ is not initialized
        set y_{pmt} as integer
        $\mathcal{U} := \mathcal{U} \cup \{(p, m, t)\}$
    solve lot sizing MIP
    foreach $(p, m, t) \in \mathcal{U}$
      fix $y_{pmt} := \bar{y}_{pmt}$
return $\bar{y}$




```
RECONSTRUCT(ȳ)
for t = 1 to T
    foreach p ∈ 𝒫
        foreach m ∈ ℳ^p
            if ȳ_pmt is not initialized
                relax y_pmt
            else
                fix y_pmt := ȳ_pmt
for t = 1 to T
    foreach p ∈ 𝒫
        𝒰 := {}
        foreach m ∈ ℳ^p
            if ȳ_pmt is not initialized
                set y_pmt as integer
                𝒰 := 𝒰 ∪ {(p, m, t)}
        solve lot sizing MIP
        foreach (p, m, t) ∈ 𝒰
            fix y_pmt := ȳ_pmt
return ȳ
```




```
RECONSTRUCT(ȳ)
for t = 1 to T
    foreach p ∈ P
        foreach m ∈ M^p
            if ȳ_pmt is not initialized
                relax y_pmt
            else
                fix y_pmt := ȳ_pmt
for t = 1 to T
    foreach p ∈ P
        U := {}
        foreach m ∈ M^p
            if ȳ_pmt is not initialized
                set y_pmt as integer
                U := U ∪ {(p, m, t)}
        solve lot sizing MIP
        foreach (p, m, t) ∈ U
            fix y_pmt := ȳ_pmt
return ȳ
```




```
RECONSTRUCT(ȳ)
for t = 1 to T
    foreach p ∈ 𝒫
        foreach m ∈ ℳ^p
            if ȳ_pmt is not initialized
                relax y_pmt
            else
                fix y_pmt := ȳ_pmt
for t = 1 to T
    foreach p ∈ 𝒫
        𝒰 := {}
        foreach m ∈ ℳ^p
            if ȳ_pmt is not initialized
                set y_pmt as integer
                𝒰 := 𝒰 ∪ {(p, m, t)}
        solve lot sizing MIP
        foreach (p, m, t) ∈ 𝒰
            fix y_pmt := ȳ_pmt
return ȳ
```

A hybrid tabu search approach

- two-fold hybrid metaheuristic for lot sizing:
  - relax-and-fix-one-product is used to initialize a solution, or complete partial solutions
  - tabu search is responsible for creating diverse points for restarting relax-and-fix.
  - before each restart:
    * the current tabu search solution is partially destructed
    * its reconstruction is made by means of relax-and-fix-one-product

Solution representation

- For tabu search:
  - variables: y_{pmt} variables
  - all continuous variables can be determined in function of these
- Thus: a tabu search solution is a matrix of $\bar{y}_{pmt}$ binary variables.

$$
\begin{aligned}
\text{minimise} \quad & z = F + V + I + B \\
\text{subject to :} \quad & F = \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} \sum_{t \in \mathcal{T}} f_{pmt} \; y_{pmt} \\
& V = \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} \sum_{t \in \mathcal{T}} v_{pmt} \; x_{pmt} \\
& I = \sum_{p \in \mathcal{P}} \sum_{t \in \mathcal{T}} i_{pt} \; h_{pt} \\
& B = \sum_{p \in \mathcal{P}} \sum_{t \in \mathcal{T}} b_{pt} \; g_{pt} \\
& h_{p,t-1} - g_{p,t-1} + \sum_{m \in \mathcal{M}^p} x_{pmt} = D_{pt} + h_{pt} - g_{pt}, \quad \forall\, p \in \mathcal{P},\ \forall\, t \in \mathcal{T} \\
& \sum_{p \in \mathcal{P} : m \in \mathcal{M}^p} \quad \frac{x_{pmt}}{\gamma_{pm}} + \tau_{pmt} \; y_{pmt} \quad \leq A_{mt}, \quad \forall\, m \in \mathcal{M},\ \forall\, t \in \mathcal{T} \\
& x_{pmt} \leq \gamma_{pm} \; A_{mt} \; y_{pmt} \quad \forall\, p \in \mathcal{P},\ \forall\, m \in \mathcal{M}^p,\ \forall\, t \in \mathcal{T} \\
& F, V, I, B \in \mathbb{R}^+ \\
& h_{pt},\ g_{pt} \in \mathbb{R}^+, \quad \forall\, p \in \mathcal{P},\ \forall\, t \in \mathcal{T} \\
& x_{pmt} \in \mathbb{R}^+,\ y_{pmt} \in \{0, 1\}, \quad \forall\, p \in \mathcal{P}, \forall\, m \in \mathcal{M},\ \forall\, t \in \mathcal{T}
\end{aligned} \tag{2}
$$

Solution evaluation

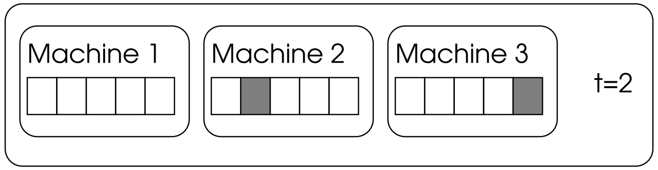
- can be made through the solution of the lot sizing model
- with all the binary variables fixed at values $\bar{y}_{pmt}$
- as all the binary variables are fixed, this problem is a linear program (LP)
- z at the optimal solution of this LP provides the evaluation of the quality of $\bar{y}_{pmt}$
- values of all the other variables x, h and g corresponding to $\bar{y}_{pmt}$ are also determined through this LP solution

Hybrid tabu search

TabuSearch(*tlim, seed, instance*)
(1) store *instance* information $\mathcal{T}, \mathcal{P}, \mathcal{M}, f, g, \ldots$
(2) initialize random number generator with *seed*
(3) $\bar{y} :=$ RelaxAndFixOneProduct()
(4) $\bar{y}^* := \bar{y}$
(5) $n := |\mathcal{T}| \times |\mathcal{P}|$
(6) $\Theta := ((-n, \ldots, -n), \ldots, (-n, \ldots, -n))$
(7) $i := 1$
(8) **while** CPUTime() $<$ *tlim*
(9) $\quad \bar{y} :=$ TabuMove($\bar{y}, \bar{y}^*, i, \Theta$)
(10) $\quad$ **if** $\bar{y}$ is better than $\bar{y}^*$
(11) $\quad\quad \bar{y}^* := \bar{y}$
(12) $\quad i := i + 1$
(13) **return** $\bar{y}^*$

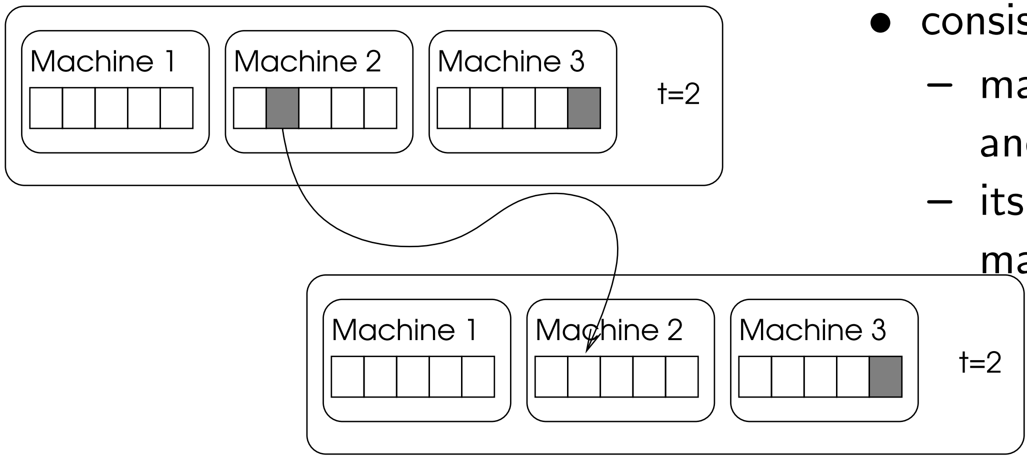
- based only on short term memory
- parameter: *tlim*, limit of CPU to be used in the search
- seed for initializing the random number generator
- name of the instance to be solved.

Neighborhood



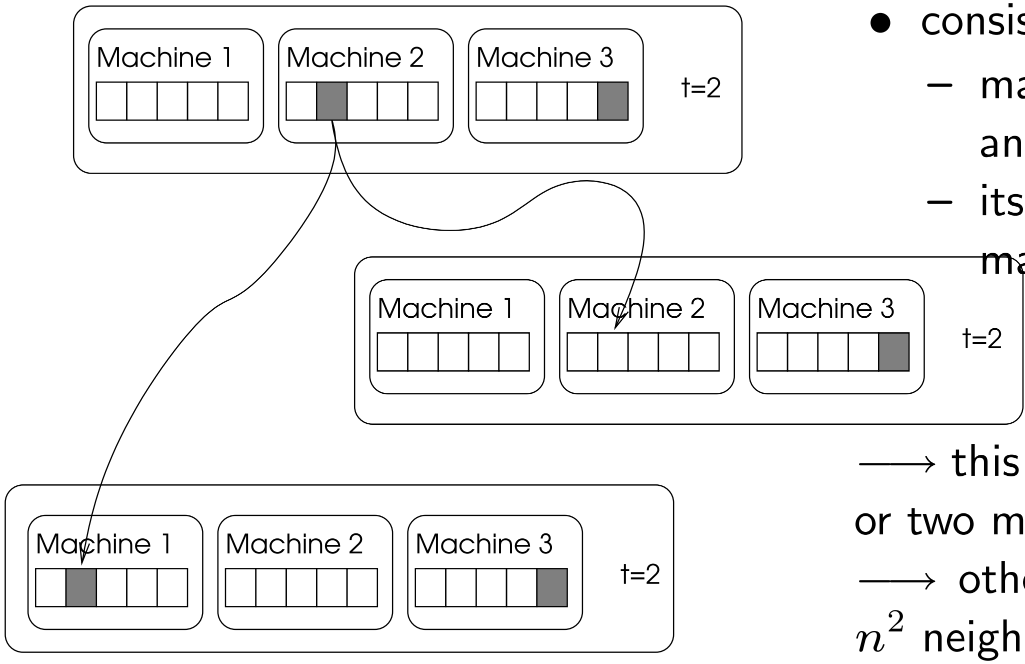
- consists of solutions where:
  - manufacturing a product in a given period and machine is stopped
  - its manufacture is attempted in different machines, on the same period

Neighborhood



- consists of solutions where:
  - manufacturing a product in a given period and machine is stopped
  - its manufacture is attempted in different machines, on the same period

Neighborhood



- consists of solutions where:
  - manufacturing a product in a given period and machine is stopped
  - its manufacture is attempted in different machines, on the same period

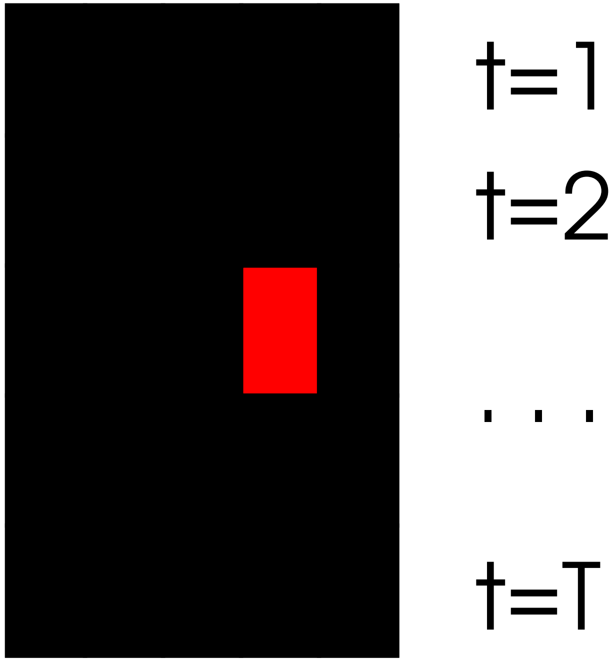
$\longrightarrow$ this is a composed neighborhood, where one or two moves are allowed.
$\longrightarrow$ otherwise: for n integer (setup) variables, n^2 neighbors to check

Tabu moves

- neighbor is returned immediately if:
  - it improves the best found solution
  - it is not tabu and it improves the input solution
- if no improving move could be found in the whole neighborhood:
  we force a **diversification:**
  - solution is partially destructed
  - best found move is then applied and made tabu
  - solution is reconstructed

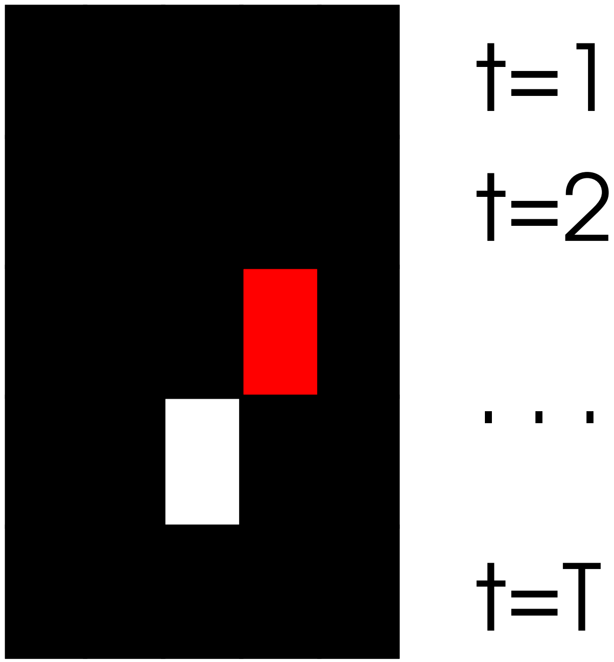
$\longrightarrow$ **hybridization** is on the destruction/reconstruction steps

Solution destruction



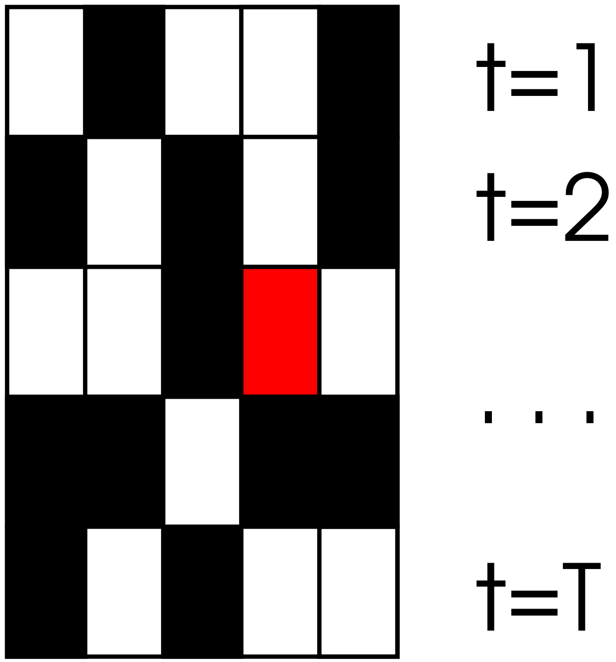
- start with a complete solution (all integer variables are fixed)
- randomly select a *non-tabu* variable

Solution destruction



- start with a complete solution (all integer variables are fixed)
- randomly select a *non-tabu* variable
- *un-initialize* it

Solution destruction



- start with a complete solution (all integer variables are fixed)
- randomly select a *non-tabu* variable
- *un-initialize* it

$\longrightarrow$ continue until having *destructed* α% of the variables

$\longrightarrow$ α is a random uniform in $[0, 1]$, drawn at each iteration

Tabu information

Tabu information: kept in the matrix Θ

Θ_{pm} holds the iteration at which a variable y_{pmt} has been updated

tabu tenure: is a random value, d

- drawn in each iteration between 1 and the number of integer variables
- if the current iteration is i, then a move involving product p and machine m:
  - is tabu if $i - \Theta_{pm} \leq d$
  - otherwise (i.e., if $i - \Theta_{pm} > d$) it is not tabu
- making it a random value simplifies the parameterization

Move during each tabu search iteration.

$\textsc{TabuMove}(\bar{y}, \bar{y}^*, i, \Theta)$

$\bar{y}' := \bar{y}$

for $t = 1$ **to** T

  foreach $p \in \mathcal{P}$

    $\mathcal{S} := \{m \in \mathcal{M}^p : \bar{y}_{pmt} = 1\}$

    $\mathcal{U} := \{m \in \mathcal{M}^p : \bar{y}_{pmt} = 0\}$

    $d := \mathscr{R}[1, |\mathcal{P}| \times |\mathcal{M}| \times |\mathcal{T}|]$

    foreach $m \in \mathcal{S}$

      fix $\bar{y}_{pmt} := 0$

      if $\bar{y}$ is better than $\bar{y}^*$ **or** $(i - \Theta_{pm} > d$ **and** $\bar{y}$ is better than $\bar{y}')$

        return $\bar{y}$

      if $i - \Theta_{pm} > d$ **and** ($\bar{y}^c$ is not initialized **or** $\bar{y}$ is better than $\bar{y}^c$)

        $\bar{y}^c := \bar{y}$, $m_1 := (p, m, t)$

      foreach $m' \in \mathcal{U}$

        fix $\bar{y}_{pm't} := 1$

        if $\bar{y}$ is better than $\bar{y}^*$ **or** $(i - \Theta_{pm} > d$ **and** $\bar{y}$ is better than $\bar{y}')$

          return $\bar{y}$

        if $i - \Theta_{pm} > d$ **and** ($\bar{y}^c$ is not initialized **or** $\bar{y}$ is better than $\bar{y}^c$)

          $\bar{y}^c := \bar{y}$, $m_1 := (p, m, t)$, $m_2 := (p, m', t)$

        restore $\bar{y}_{pm't} := 0$

      restore $\bar{y}_{pmt} := 1$

$\alpha := \mathscr{R}$
un-initialize α% of the $\bar{y}^C$ variables
if $\bar{y}^C$ is not initialized
    select a random index (p, m, t)
    $\bar{y}^C := \bar{y},\ \bar{y}^C_{pmt} := 1 - \bar{y}_{pmt},\ \Theta_{pm} := i$
else
    $(p, m, t) := m_1,\ \Theta_{pm} := i$
    if m_2 is initialized
        $(p, m, t) := m_2,\ \Theta_{pm} := i$
$\bar{y} :=$ $\textsc{Reconstruct}(\bar{y}^C)$
return $\bar{y}$

Computational results

- algorithms were tested on a series of benchmark instances
- instances derived from a real-world problem
  - 12 products
  - 12 periods
  - 15 machines
  - random demand (average: true estimated demand)
- smaller instances:
  - reduce the number of periods
  - randomly select a subset of products
  - machines: those compatible with the selected products

Instances – practical benchmarks

Name	Number of periods	Number of products	Number of integers	Number of variables	Number of constraints
inst-02	2	2	20	56	45
inst-05	5	5	135	334	235
inst-07	7	7	210	536	369
inst-09	9	9	306	796	554
inst-12	12	12	492	1300	857

Results – practical benchmarks

Name	Relax-and-fix		Hybrid tabu search sol.			branch-and-bound best sol.
	time (s)	solution	worst	average	best	
inst-02	< 1	13.897	13.897	13.897	13.897	13.897*
inst-05	1.8	50.536	48.878	48.878	48.878	48.878
inst-07	2.9	131.095	126.030	126.265	126.595	127.604
inst-09	5.6	213.981	207.441	208.206	208.841	235.125
inst-12	13.1	277.451	274.283	274.397	274.626	431.660

Hybrid tabu search and branch-and-bound: 3600 seconds CPU time

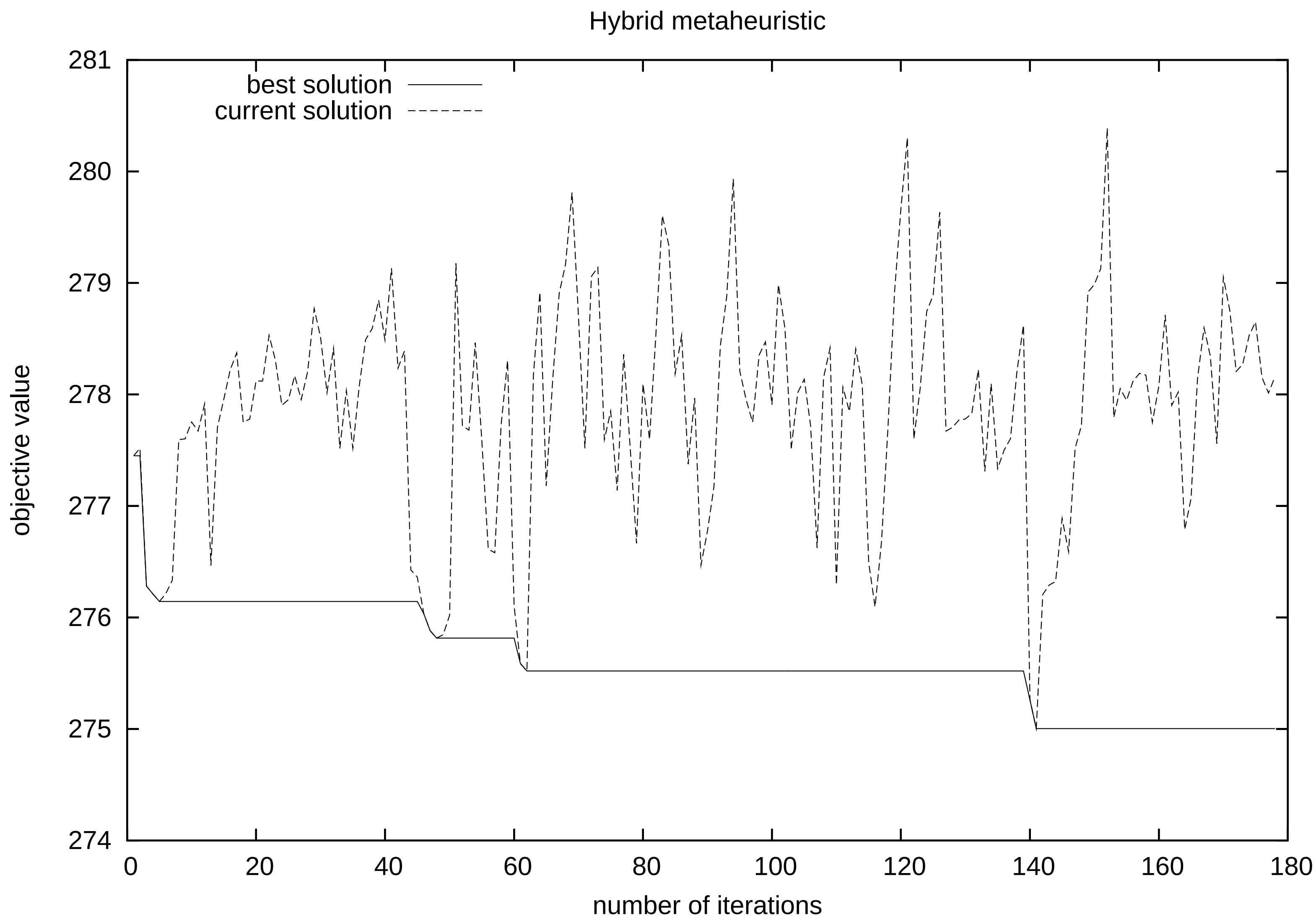
Results – LOTSIZELIB benchmarks

Name	Relax-and-fix (average)		Hybrid tabu search sol.			branch-and-	optimal
	time (s)	solution	worst	average	best	bound best sol.	solution
pp08a	<1	7638.0	7380	7374	7360	7350	7350
rgna	<1	82.2	82.2	82.2	82.2	82.2	82.2
set1ch	13.4	56024.3	55243.5	55089.6	54950	60517.7	54537
tr6-15	1.3	40767.6	38357	38238	38054	39388	37721
tr6-30	5.1	67057.0	63422	63246.2	63132	63711	61746^*
tr12-30	69.1	143014.0	137371	136762.8	136299	1940337	130599^*

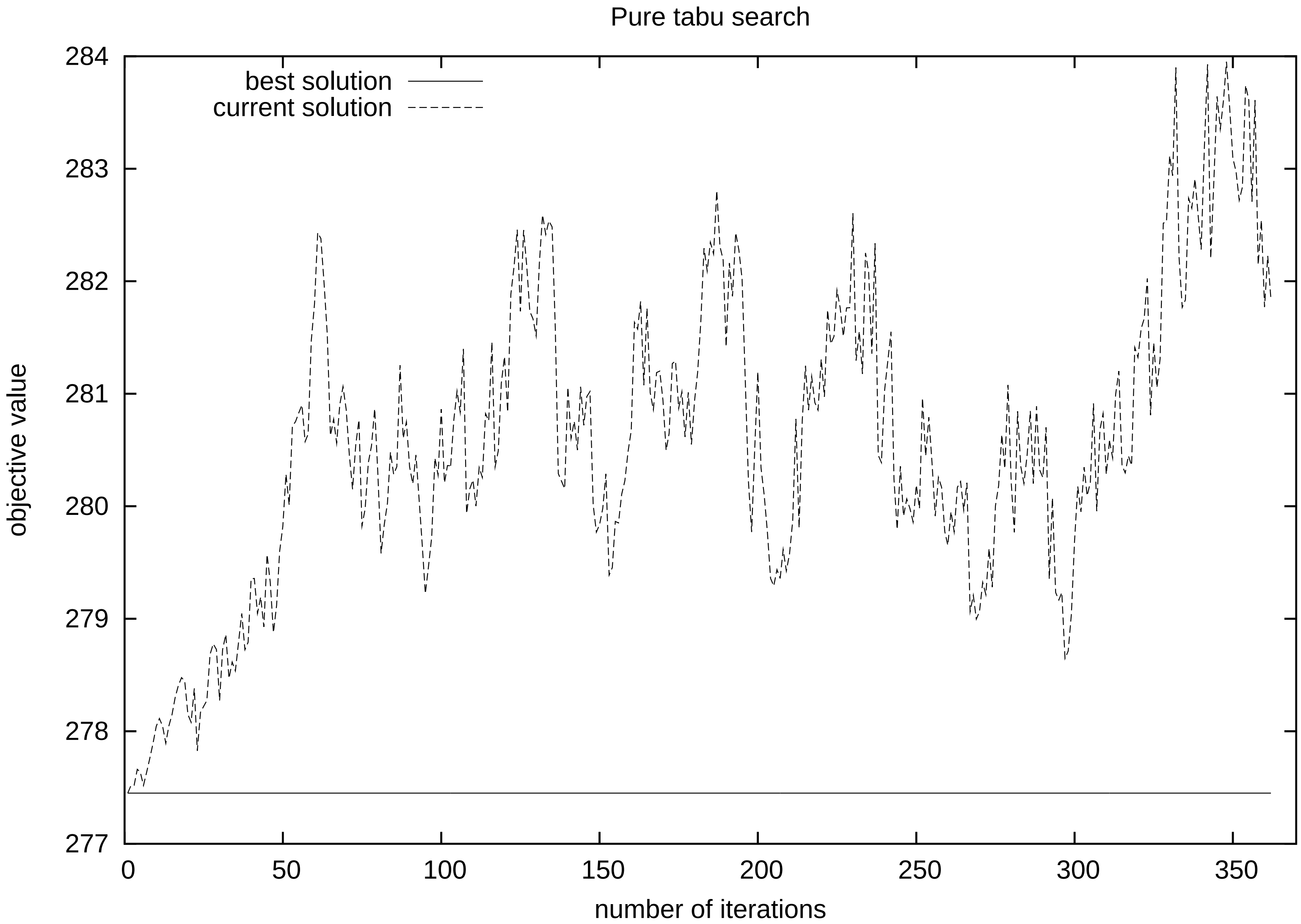
Hybrid tabu search and branch-and-bound: 3600 seconds CPU time

Optimal solutions: as reported in LOTSIZELIB.

(* indicate best known solutions)



Hybrid metaheuristic
best solution
current solution
objective value
number of iterations
281
280
279
278
277
276
275
274
0
20
40
60
80
100
120
140
160
180


Pure tabu search
best solution
current solution
objective value
277
278
279
280
281
282
283
284
0
50
100
150
200
250
300
350
number of iterations

Conclusion

- main motivation for this work
  - solve practical problem
  - exploitation of relax-and-fix in a setup which enforced diversity
  - tabu search mechanism was responsible for imposing changes on the solution
  - after changes were made:
    - a part of the solution (not involving the latest changes) was destructed
    - relax-and-fix was used to rebuild it.
- why hybridize:
  - non-improving moves made by tabu search rapidly force the solution into rather poor regions
  - reason: large number of moves required to change good solution into another good solution
  - "moves" done by relax-and-fix whenever tabu search cannot find improving neighbor
  - when improving neighbors are found, the destruction/reconstruction cycle were skipped
- computational results show advantage of this strategy, as compared to:
  - simple relax-and-fix-one-product heuristic
  - time-limited branch-and-bound