A GRASP approach for the machine reassignment problem

Michaël Gabay
Sofia Zaourar

Laboratoire G-SCOP
LJK - Inria Grenoble
France

Team J19 - Open source

July 10, 2012
Summary

1 The problem
   - Overview
   - Constraints
   - Costs

2 Our solution
   - Scheme
   - Vector bin packing
   - Another approach
   - Local search

3 Final solution
   - Solution
   - Results
A GRASP approach for the machine reassignment problem

The problem

Overview
A GRASP approach for the machine reassignment problem

The problem

Overview
A GRASP approach for the machine reassignment problem

The problem

Overview
Constraints

- Capacity constraints and transient usage constraints
- Conflict constraints
- Spread constraints
- Dependency constraints
Capacity constraints
Capacity constraints
Capacity constraints
Capacity constraints
A GRASP approach for the machine reassignment problem

The problem

Constraints

Capacity constraints
Capacity constraints

Vector bin packing problem, Garey et al., 76
Dependency constraints

\( s_1 \) depends on \( s_2 \):

\[
\begin{align*}
  &P_1, s_1 \\
  &P_2, s_2 \\
  &M_1 \\
  &N_1 \\
  &P_3, s_3 \\
  &M_2 \\
  &N_2 \\
  &P_4, s_2 \\
  &M_3 \\
  &N_3 \\
  &P_5, s_3 \\
  &M_4
\end{align*}
\]
Dependency constraints

$s_1$ depends on $s_2$:
Dependency constraints

$s_1$ depends on $s_2$:
Dependency constraints

Idea: break up the problem by neighborhoods

⇒ smaller subproblems with no dependency constraints.

Solvable using integer programming ⇒ Matheuristic approach.
The problem

Costs

Costs

- Process / Service / Machine *move* costs
- Load costs
- *Balance* costs
A GRASP approach for the machine reassignment problem

The problem

Costs

Load costs
Load costs

Use load costs to **guide** search
A GRASP approach for the machine reassignment problem

Our solution

Scheme

GRASP?

GRASP = Greedy randomized adaptive search procedure

Optimization scheme:

1. Create a solution using vector bin packing heuristic
2. Optimize it with a local search
3. Goto 1
Vector Bin Packing

∀m ∈ M, ∀p ∈ P, volume v:

\[ v(m) = \sum_{r \in R} \frac{C(m, r)}{C(r)} \], \quad \forall p \in P, \quad v(p) = \sum_{r \in R} \frac{R(p, r)}{R(r)} \]
A GRASP approach for the machine reassignment problem

Our solution

Vector bin packing

Vector Bin Packing

∀m ∈ M, ∀p ∈ P, volume v:

\[ v(m) = \sum_{r \in R} \frac{C(m, r)}{C(r)}, \quad v(p) = \sum_{r \in R} \frac{R(p, r)}{R(r)} \]

Greedy : First Fit Decreasing (FFD)

**FFD Bin-centric** heuristic (Panigrahy et al., 2011):

1. Sort machines by increasing \( v(m) \), processes by decreasing \( v(p) \)
2. Pop the smallest remaining machine \( m \)
3. While some processes fit into \( m \), place the largest remaining one
4. Goto 2
A GRASP approach for the machine reassignment problem

Our solution

Vector Bin Packing

∀m ∈ M, ∀p ∈ P, volume v:

\[
v(m) = \sum_{r \in R} \frac{C(m, r)}{C(r)}, \quad v(p) = \sum_{r \in R} \frac{R(p, r)}{R(r)}\]

Greedy : First Fit (FF)

**FF Bin-Balancing** (item-centric) heuristic:

1. Sort machines by increasing v(m), processes by decreasing v(p), i = 1
2. Pop the largest remaining process p
3. While some machine can host p, place p on the smallest \( j \geq i + 1 \) (with a cyclic order), \( i = j \)
4. Goto 2
A GRASP approach for the machine reassignment problem

Our solution

Vector bin packing

**Vector Bin Packing**

\[ \forall m \in M, \forall p \in P, \text{ volume } v: \]

\[ v(m) = \sum_{r \in R} \frac{C(m, r)}{C(r)}, \quad v(p) = \sum_{r \in R} \frac{R(p, r)}{R(r)} \]

Greedy Random

**FF Mixed orderings** heuristic:

1. Sort machines by increasing \( v(m) \), processes by decreasing \( v(p) \), or random sort both
2. Run FFD Bin-centric, run FF Bin-Balancing
3. Goto 1
A GRASP approach for the machine reassignment problem

Our solution

Vector bin packing

Vector Bin Packing

Constraints verified?

- Capacity and transient usage constraints
- Conflict constraints
- Spread constraints
- Dependency constraints
A GRASP approach for the machine reassignment problem

Our solution

Vector bin packing

Vector Bin Packing

Constraints verified?

- Capacity and transient usage constraints ⇒ YES
- Conflict constraints ⇒ YES
- Spread constraints
- Dependency constraints
A GRASP approach for the machine reassignment problem

Our solution: Vector bin packing

Vector Bin Packing

Constraints verified?

- Capacity and transient usage constraints \(\Rightarrow\) YES
- Conflict constraints \(\Rightarrow\) YES
- Spread constraints
- Dependency constraints \(\Rightarrow\) YES - Assign all processes of a neighborhood to the same neighborhood (possibly not the original one)
A GRASP approach for the machine reassignment problem

Our solution

Vector bin packing

Vector Bin Packing

Constraints verified ?

- Capacity and transient usage constraints ⇒ YES
- Conflict constraints ⇒ YES
- Spread constraints ⇒ REPAIR
- Dependency constraints ⇒ YES - Assign all processes of a neighborhood to the same neighborhood (possibly not the original one)
A GRASP approach for the machine reassignment problem

Our solution

Vector bin packing

GRASP

1. Use FF Mixed orderings to get a feasible solution
2. Local search on the solution
3. Goto 1
A GRASP approach for the machine reassignment problem

Our solution

Vector bin packing

Problems with VBP

VBP heuristics are very fast but...

- Solutions’ costs are too high
- Not so interesting when processes do not change neighborhood,... nor when they all change...
- 10 B instances, 6 feasible,... all violate spread constraints
- Repairing may be too difficult / time consuming
A matheuristic

Idea:

1. For each *neighborhood*, solve the assignment problem using an IP (smaller subproblem with no dependency constraints)
2. Optimize using local search
3. Goto 1

(*implemented using* Coin-Osi/Clp/Cbc)
A GRASP approach for the machine reassignment problem

Our solution

Another approach

A matheuristic: Problems
A matheuristic: Problems

Subproblems are still too big
A matheuristic: Problems

Subproblems are still too big

Solution:

- Divide into smaller subproblems
- Randomly / Guided by load cost (most expensive machines with cheapest w.r.t. load costs)
A matheuristic: Problems

Tune size parameters:
Compromise between time consumption and feasibility
A matheuristic: Problems

Tune size parameters:

Compromise between time consumption and feasibility

Solution:

- Set maximum number of machines and processes to be considered at once
- Set maximum number of nodes
A matheuristic: Problems

Pure local search is better
A GRASP approach for the machine reassignment problem

Our solution

Another approach

A matheuristic: Problems

Pure local search is better (and deadline approaches)
A GRASP approach for the machine reassignment problem

Our solution

Another approach

A matheuristic: Problems

Pure local search is better (and deadline approaches)

Solution:

Forget about the Matheuristic :-(

Local search

Two simple moves:

- Move $p$ from $m_1$ to $m_2$
- Swap $p_1$ and $p_2$ on $m_1$ and $m_2$

Efficient Structures + randomization
Local search

Two simple moves:

- Move $p$ from $m_1$ to $m_2$
- Swap $p_1$ and $p_2$ on $m_1$ and $m_2$

Efficient Structures + randomization

Hill Climbing: if a move is feasible and decreases the total cost, do it!
Local search

Two simple moves:
- Move $p$ from $m_1$ to $m_2$
- Swap $p_1$ and $p_2$ on $m_1$ and $m_2$

Efficient Structures + randomization

Hill Climbing: if a move is feasible and decreases the total cost, do it!

Guided first step: Move processes from the machines with the highest load costs to cheaper machines
Our solution
Local search

Local search

Two simple moves:
- Move \( p \) from \( m_1 \) to \( m_2 \)
- Swap \( p_1 \) and \( p_2 \) on \( m_1 \) and \( m_2 \)

Efficient Structures + randomization

Hill Climbing: if a move is feasible and decreases the total cost, do it!

Guided first step: Move processes from the machines with the highest load costs to cheaper machines

Blocked ? Restart from a previous solution with a new seed
Final solution

Local search

Two parallel local search with different strategies
Final solution

Local search

Two parallel local search with different strategies

Open source under LGPL v3 license, available at:

https://github.com/TeamJ19ROADEF2012

(Includes VBP and the matheuristic)
## Final solution

### Results

<table>
<thead>
<tr>
<th>Instance</th>
<th>Cost</th>
<th>Real time</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>B 1</td>
<td>3 609 228 327</td>
<td>4m58.5s</td>
<td>9m53.4s</td>
</tr>
<tr>
<td>B 2</td>
<td>1 017 459 868</td>
<td>4m58.5s</td>
<td>9m54.4s</td>
</tr>
<tr>
<td>B 3</td>
<td>170 139 317</td>
<td>4m58.5s</td>
<td>9m53.4s</td>
</tr>
<tr>
<td>B 4</td>
<td>4 677 960 720</td>
<td>4m58.5s</td>
<td>9m54.2s</td>
</tr>
<tr>
<td>B 5</td>
<td>930 031 137</td>
<td>4m58.5s</td>
<td>9m53.8s</td>
</tr>
<tr>
<td>B 6</td>
<td>9 525 886 513</td>
<td>4m58.5s</td>
<td>9m53.7s</td>
</tr>
<tr>
<td>B 7</td>
<td>14 911 018 492</td>
<td>4m58.5s</td>
<td>9m52.4s</td>
</tr>
<tr>
<td>B 8</td>
<td>1 217 854 951</td>
<td>4m58.5s</td>
<td>9m53.0s</td>
</tr>
<tr>
<td>B 9</td>
<td>15 886 884 119</td>
<td>4m58.5s</td>
<td>9m54.1s</td>
</tr>
<tr>
<td>B 10</td>
<td>18 391 528 354</td>
<td>4m58.5s</td>
<td>9m51.0s</td>
</tr>
</tbody>
</table>
Questions ?