Training a neural network to select dispatching rules in real time

Wiem Mouelhi-Chibani, Henri Pierreval


2019. 07. 31
Jeong Hoon Shin
Contents

1. Introduction
2. Selecting efficient dispatching rules: related research
3. Problem formulation
4. Proposed method
5. Example: application to a simplified flow-shop
6. Conclusion
1. Introduction
Introduction

- Dispatching Rule (DR)
  - Definition: rules used to select the next job to process from jobs awaiting service.
  - there is no DR that is globally better than all the others.
  - efficiency depends on the characteristics of the system (the operating condition parameters and the production objectives)

- changing dynamically the DR (when, and which DR)
  - can yield better results.
  - Artificial intelligence (AI) can be very useful.

- This study suggests a new approach
  - learns without any training set
    - unsupervised learning
2. Selecting efficient dispatching rules: related research
Related research

- Boukachour, 1992
  - Performance of DRs greatly depends on the configuration of the studied system, the operating conditions and the performance criterion used to evaluate the DRs

- Finding best rule in steady state simulations
  - Single-pass period method: single long period of time to study the DR performance
  - Statistical methods or learning methods (clustering, segmentation, machine learning)

- For dynamic and real scheduling.
  - Multi-pass period method: Decompose simulation time into small periods
  - Few papers use learning methods
    - the results can be very sensitive to the period length.
    - a succession of good local strategies does not necessarily yield a good global performance.
Related research & approach of this study

- Directly consider the dynamic changes of the system state to select DRs.
  - do not decompose time into periods for DR selection.
  - Consider triggering events, expert system (simulation optimization)

- In learning method
  - A training set from real time environment is quite difficult to build.

- The approach in this study
  - dynamic (no decomposition of time in periods): according to the system state.
  - allows knowledge about how to decide in real time
    - determining which dispatching rule (DR) to select when a triggering event occurs.
    - using Neural Network (NN)
3. Problem formulation
Notations & Problem

Notations

- set $M$ of $m$ machines: $M = \{m_1, m_2, \ldots, m_m\}$.
- set of $r$ candidate dispatching rules that can be used for scheduling machine $m_j$: $DR_j = \{DR_{j1}, DR_{j2}, \ldots, DR_{jr}\}$.
- vector of system parameters: $D = \{D_1, D_2, \ldots, D_l\}$.
- vector of state variables: $S(t) = \{S_1(t), S_2(t), \ldots, S_k(t)\}$. (# of jobs in Queue, # of idle resources in given WorkCenter)

Objective: to select, at given instants $t$ (triggering event time), the most suitable dispatching rule $DR_{jr}$, to determine the next job to process on a given machine $m_j$, so that the expectation of the performance $E(f(\Sigma))$, computed on the study period, is optimized.

Problem

- to determine for the period of time for which the system is studied the right sequence of dispatching rules $\Sigma$ to optimize the expectation of the performance function $E(f(\Sigma))$. 

4. Proposed method
**NN configuration & simulation module**

- **Neural Network**
  - Output: # of node = # of decision points (select DR) in system.
  - Learning = determining the values of $W = \{w_1, w_2, ..., w_i\}$.

- **Simulation module**
  - Decision module: learned NN
  - Physical module: construct using simulation tool

---

**Fig. 2. Neural network for selecting dispatching rules**

**Fig. 3. Simulation module of the workshop**
Training NN using Local search

- **Solution**: $W = \{w_1, w_2, ..., w_i\}$. (initial solution : $W_0$ random choose)
- **Objective**: optimize $E(f(W))$.
- **Complexity**: relies on # of weights ($i$ in $W = \{w_1, w_2, ..., w_i\}$)

Summary of the training methodology

**Step 1**: Define the performance criteria for the dynamic scheduling. Identify the system variables that may influence the decisions ($D$ and $S$). Identify the set of DR that can be relevant on each machine.

**Step 2**: Build a simulation model of the manufacturing system. Build a neural network. Connect the simulation and the NN so that the system state and parameters and the DR chosen can be communicated.

**Step 3**: Select a simulation optimization approach. Use the NN weights (adjusted parameters) as decision variables for the optimization. Use the simulation to measure the objective function.

**Step 4**: Run simulation optimization. Collect the resulting NN configuration.
5. Example: application to a simplified flow-shop
Each machine is periodically out of service for changing tools.

Process times of each WC ~ Exp Distribution

Due date: assigned at the time of arrival at the shop. \( D_i = A_i + \alpha \sum p_{ij} \) \( (1) \)

Shop load (\%) = \( \beta (t_1 + t_2)/4 \) \( (2) \)

Used rules: FAS, FIQ, EDD, SPT, and LPT

In Barrett and Barman (1986), best results: combinations of SPT and EDD (SPT–SPT, SPT–EDD, EDD–EDD)
Model, NN

- Simulation model (using arena)
  - Run length: 15,000h (4yr WT)
  - Shop empty at the beginning
  - Warm up period: 7000h
  - Simulation replicated & averaged

- NN (VC++)
  - System state variables $S_{19}$ to $S_{22}$ have been empirically determined after preliminary simulation runs.
  - Candidate DRs (output layer)
    - $DR_1 = \{FIQ, EDD, SPT, LPT\}$
    - $DR_2 = \{FIQ, FIS, EDD, SPT, LPT\}$
  - 168 weights: $22(\text{state var}) \times 7(\text{hidden}) + 7 \times 2(\text{output})$

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)} \quad (3)$$

Table 1: System parameters.

<table>
<thead>
<tr>
<th>$D_1$</th>
<th>Shop load: 91%</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_2$</td>
<td>Processing times variation (addition of a deviation normally distributed $N(0, 0.3)$)</td>
</tr>
<tr>
<td>$D_3$</td>
<td>Minimum job process time: 0.1 units of time</td>
</tr>
<tr>
<td>$D_4$</td>
<td>Allowance factor $\alpha$ fixed at 3</td>
</tr>
</tbody>
</table>

Table 2: System state variables.

| $S_1, S_{11}$ | Number of jobs, respectively, in the first and the second queue |
| $S_2, S_3, S_4$ | Minimum, maximum and average process time on WC1 of the jobs waiting in the first queue |
| $S_5, S_6, S_7$ | Minimum, maximum and average process time on WC2 of the jobs waiting in the first queue |
| $S_8, S_9, S_{10}$ | Minimum, maximum and average process time on WC2 of jobs waiting in the second queue |
| $S_{12}, S_{13}$ | Minimum, maximum and average slack time of jobs waiting in the first queue |
| $S_{14}$ | Minimum, maximum and average slack time of jobs waiting in the second queue |
| $S_{15}, S_{16}$ | Number of the concerned waiting queue |
| $S_{17}$ | Percentage of jobs in the concerned waiting queue with a process time less than 1.5 units of time |
| $S_{20}$ | Percentage of jobs in the concerned waiting queue with a process time between 1.5 and 3 units of time |
| $S_{21}$ | Percentage of jobs in the concerned waiting queue with a process time between 3 and 4.5 units of time |
| $S_{22}$ | Percentage of jobs in the concerned waiting queue with a process time greater than 4.5 units of time |
NN Opt & Exp

- NN optimize (using SA).

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial temperature</td>
<td>7000</td>
</tr>
<tr>
<td>Cooling ratio</td>
<td>0.8</td>
</tr>
<tr>
<td>$\sigma_{\text{max}}$</td>
<td>20</td>
</tr>
<tr>
<td>$\sigma_{\text{min}}$</td>
<td>5</td>
</tr>
<tr>
<td>The total number of weights</td>
<td>168</td>
</tr>
<tr>
<td>$p_{\text{max}}$</td>
<td>20</td>
</tr>
<tr>
<td>Stopping threshold, $\varepsilon$</td>
<td>0.001</td>
</tr>
</tbody>
</table>

The perturbation of solutions (in SA)

\[ g(l) = E[1 + ( (a - 1) \times e^{-l^2/T_0 \times 10} )] \]  \hspace{1cm} (4)

\[ s(T_l) = \left[ \frac{(\sigma_{\text{max}} - \sigma_{\text{min}})}{(T_0 - \varepsilon)} \right] \times (T_l - T_0) + \sigma_{\text{max}} \]  \hspace{1cm} (5)

- Experiment & results.
  - 10 replications for each simulation run (cpu time)
  - 50 replications to compare Barrett and Barman (1986)
  - simulation period of 4yrs (55h on a PC)
  - SPT-EDD & SPT-SPT yield good results
  - NN with opt weights compete SPT-SPT

<table>
<thead>
<tr>
<th>DR selected</th>
<th>Mean tardiness</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDD--EDD</td>
<td>11.99342</td>
</tr>
<tr>
<td>EDD--SPT</td>
<td>7.27360</td>
</tr>
<tr>
<td>SPT--EDD</td>
<td>6.67160</td>
</tr>
<tr>
<td>SPT--FIFO</td>
<td>9.33060</td>
</tr>
<tr>
<td>SPT--SPT</td>
<td>3.69815</td>
</tr>
<tr>
<td>EDD--FIFO</td>
<td>14.00025</td>
</tr>
<tr>
<td>Initial NN</td>
<td>25.43</td>
</tr>
<tr>
<td>Final NN</td>
<td>3.44712</td>
</tr>
</tbody>
</table>
6. Conclusion
Conclusion & Further research

- Proposed a new approach
  - To solve dynamic scheduling in real time DR selection environment.
  - depending on the workshop characteristics and the system state

- Implemented
  - Simulation : for the problem of Barrett and Barman (1986)
  - NN : determine weights using local search(SA)

- results
  - The result of NN with optimized weight compete the best result of Fixed rule(SPT-SPT).

- Further research.
  - use a distributed simulation optimization approach. (reduce cpu time)
  - Search : better method. good initial weights.
  - more complex problems
감사합니다