Evolutionary learning of weighted linear composite dispatching rules for scheduling
Case study for JSP and PFSP

Helga Ingimundardóttir
Thomas Philip Runarsson

University of Iceland

October, 2014
Outline

Introduction

Job Shop Scheduling

Evolutionary search with CMA-ES

Experiments

Summary and conclusions
Introduction

Job Shop Scheduling

Evolutionary search with CMA-ES

Experiments

Summary and conclusions
Motivation

General Goal

General goal is how to search for good solutions for an arbitrary problem domain.

Automate the design of optimization algorithms.

Use of randomly sampled problem instances and their corresponding optimal vs. suboptimal solutions.
Case Study: JSP and PFSP

Abstract

Framework for creating dispatching rules for JSP and PFSP. Linear classification to identify good dispatches from worse ones. Robust for higher dimensions.

Keywords: Scheduling ● Composite dispatching rules ● JSP ● PFSP ● Evolutionary Search ● Performance Measures ● Scalability
Introduction

Job Shop Scheduling

Evolutionary search with CMA-ES

Experiments

Summary and conclusions
Job Shop Scheduling (1)

JSP

Simple job shop scheduling problem is where \( n \) jobs are scheduled on a set of \( m \) machines, subject to constraints:

- each job must follow a predefined machine order,
- that a machine can handle at most one job at a time.

**Objective:** schedule the jobs so as to minimize the maximum completion time, i.e., makespan, \( C_{\text{max}} \).

PFSP

Permutation flow shop scheduling is the same as JSP except the predefined machine order is homogeneous for all jobs.
## Job Shop Scheduling (2)

### Problem space distributions used in experimental studies

<table>
<thead>
<tr>
<th>name</th>
<th>size</th>
<th>$N_{\text{train}}$</th>
<th>$N_{\text{test}}$</th>
<th>note</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{6 \times 5}^{f \text{. rnd}}$</td>
<td>6 × 5</td>
<td>500</td>
<td>–</td>
<td>random</td>
</tr>
<tr>
<td>$P_{6 \times 5}^{f \text{. rndn}}$</td>
<td>6 × 5</td>
<td>500</td>
<td>–</td>
<td>random-narrow</td>
</tr>
<tr>
<td>$P_{6 \times 5}^{f \text{. jc}}$</td>
<td>6 × 5</td>
<td>500</td>
<td>–</td>
<td>job-correlated</td>
</tr>
<tr>
<td>$P_{10 \times 10}^{f \text{. rnd}}$</td>
<td>10 × 10</td>
<td>–</td>
<td>500</td>
<td>random</td>
</tr>
<tr>
<td>$P_{10 \times 10}^{f \text{. rndn}}$</td>
<td>10 × 10</td>
<td>–</td>
<td>500</td>
<td>random-narrow</td>
</tr>
<tr>
<td>$P_{10 \times 10}^{f \text{. jc}}$</td>
<td>10 × 10</td>
<td>–</td>
<td>500</td>
<td>job-correlated</td>
</tr>
<tr>
<td>$P_{6 \times 5}^{j \text{. rnd}}$</td>
<td>6 × 5</td>
<td>500</td>
<td>–</td>
<td>random</td>
</tr>
<tr>
<td>$P_{6 \times 5}^{j \text{. rndn}}$</td>
<td>6 × 5</td>
<td>500</td>
<td>–</td>
<td>random-narrow</td>
</tr>
<tr>
<td>$P_{10 \times 10}^{j \text{. rnd}}$</td>
<td>10 × 10</td>
<td>–</td>
<td>500</td>
<td>random</td>
</tr>
<tr>
<td>$P_{10 \times 10}^{j \text{. rndn}}$</td>
<td>10 × 10</td>
<td>–</td>
<td>500</td>
<td>random-narrow</td>
</tr>
</tbody>
</table>
Dispatching rules (DR) for constructing JSSP

Starts with an empty schedule and adds on one job at a time. When a machine is free the DR inspects the waiting/available jobs and selects the job with the highest priority.

Complete schedule consists of $\ell = n \times m$ sequential dispatches. At each dispatch $k$ features $\phi(k)$ for the temporal schedule are calculated.

Performance of DR is compared with its optimal makespan, as percentage relative deviation from optimality: $\rho = \frac{C_{\text{max}}^{DR} - C_{\text{max}}^{\text{opt}}}{C_{\text{max}}^{\text{opt}}} \cdot 100\%$
### Features for JSSP

<table>
<thead>
<tr>
<th>$\phi$</th>
<th>Feature description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi_1$</td>
<td>processing time for job on machine</td>
</tr>
<tr>
<td>$\phi_2$</td>
<td>start-time</td>
</tr>
<tr>
<td>$\phi_3$</td>
<td>end-time</td>
</tr>
<tr>
<td>$\phi_4$</td>
<td>when machine is next free</td>
</tr>
<tr>
<td>$\phi_5$</td>
<td>current makespan</td>
</tr>
<tr>
<td>$\phi_6$</td>
<td>work remaining</td>
</tr>
<tr>
<td>$\phi_7$</td>
<td>most work remaining</td>
</tr>
<tr>
<td>$\phi_8$</td>
<td>slack time for this particular machine</td>
</tr>
<tr>
<td>$\phi_9$</td>
<td>slack time for all machines</td>
</tr>
<tr>
<td>$\phi_{10}$</td>
<td>slack time weighted w.r.t. number of operations already assigned</td>
</tr>
<tr>
<td>$\phi_{11}$</td>
<td>time job had to wait</td>
</tr>
<tr>
<td>$\phi_{12}$</td>
<td>size of slot created by assignment</td>
</tr>
<tr>
<td>$\phi_{13}$</td>
<td>total processing time for job</td>
</tr>
</tbody>
</table>
A schedule being built at step $k = 16$. The dashed boxes represent five different possible jobs that could be scheduled next using a DR.
Introduction

Job Shop Scheduling

Evolutionary search with CMA-ES

Experiments

Summary and conclusions
Evolutionary search

Instead of using logistic regression for to find the weights $w$ for linear preference function:

$$h(\phi) = \sum_{i=1}^{d} w_i \phi = \langle w \cdot \phi \rangle.$$ 

a widely-used evolutionary algorithm, Covariance Matrix Adaptation Evolution Strategy (CMA-ES), is applied directly on the desired objective function. For this study both, a) expected relative error, $E[\rho]$; and b) final makespan, $E[C_{\text{max}}]$, will be considered.

**Benefit** No need to collect training data beforehand.

**Drawback** Computationally expensive to evaluate $E[\cdot]$.
Introduction

Job Shop Scheduling

Evolutionary search with CMA-ES

Experiments

Summary and conclusions
Experiments (1)

Fitness for optimising with CMA-ES

Box-plot of training data for percentage relative deviation from optimality, $\rho$, when implementing the final weights obtained from CMA-ES optimisation, using both obj. functions, $\mathbb{E}[C_{\text{max}}]$ and $\mathbb{E}[\rho]$. 
Experiments (2)

Fitness for optimising with CMA-ES

Fitness for minimising w.r.t. $C_{\text{max}}$ and $\rho$, per generation of CMA-ES.
Experiments (3)

Evolution of weights of features

Evolution of weights of features at each generation of the CMA-ES optimisation, for minimisation w.r.t. $C_{\text{max}}$ (left) and $\rho$ (right).
Introduction

Job Shop Scheduling

Evolutionary search with CMA-ES

Experiments

Summary and conclusions
Summary and conclusions (1)

Introduced a framework for learning linear composite dispatch rules for scheduling.

The approach finds linear weights by direct optimisation with CMA-ES.

The method significantly outperforms SDRs from the literature, and authors’ previous work which were based on preference learning.
Summary and conclusions (2)

CMA-ES optimisation

Benefits:

- Does not rely on optimal solutions – although if known, they can help getting out of local minima.
- Scalable – model based on $6 \times 5$ successfully applied to $10 \times 10$.

Drawbacks:

- Computationally expensive.
- Limited to linear preference function $h(\cdot)$.
Summary and conclusions (3)

Future work

Facilitate evolutionary search by use of surrogate models which indirectly estimate mean expected error w.r.t. current population without loss in performance
Thank you for your attention

Questions?

Contact: Helga Ingimundardóttir, hei2@hi.is