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

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

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, $C_{\max}$.

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

name	size	N_{train}	N_{test}	note	
PFSP	$\mathcal{P}^{6\times5}_{f.rnd}$	6×5	500	–	random
	$\mathcal{P}^{6\times5}_{f.rndn}$	6×5	500	–	random-narrow
	$\mathcal{P}^{6\times5}_{f.jc}$	6×5	500	–	job-correlated
	$\mathcal{P}^{10\times10}_{f.rnd}$	10×10	–	500	random
	$\mathcal{P}^{10\times10}_{f.rndn}$	10×10	–	500	random-narrow
	$\mathcal{P}^{10\times10}_{f.jc}$	10×10	–	500	job-correlated
JSP	$\mathcal{P}^{6\times5}_{j.rnd}$	6×5	500	–	random
	$\mathcal{P}^{6\times5}_{j.rndn}$	6×5	500	–	random-narrow
	$\mathcal{P}^{10\times10}_{j.rnd}$	10×10	–	500	random
	$\mathcal{P}^{10\times10}_{j.rndn}$	10×10	–	500	random-narrow

Job Shop Scheduling (3)

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: $\rho = \frac{C_{\max}^{DR} - C_{\max}^{opt}}{C_{\max}^{opt}} \cdot 100\%$

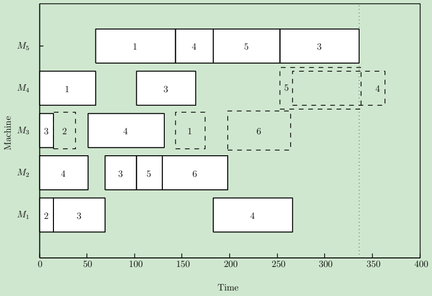
Job Shop Scheduling (4)

Features for JSSP

ϕ	Feature description
ϕ_1	processing time for job on machine
ϕ_2	start-time
ϕ_3	end-time
ϕ_4	when machine is next free
ϕ_5	current makespan
ϕ_6	work remaining
ϕ_7	most work remaining
ϕ_8	slack time for this particular machine
ϕ_9	slack time for all machines
ϕ_{10}	slack time weighted w.r.t. number of operations already assigned
ϕ_{11}	time job had to wait
ϕ_{12}	size of slot created by assignment
ϕ_{13}	total processing time for job

Job Shop Scheduling

Example



A schedule being built at step $k = 16$. The dashed boxes represent five different possible jobs that could be scheduled next using a DR.

Evolutionary search

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

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

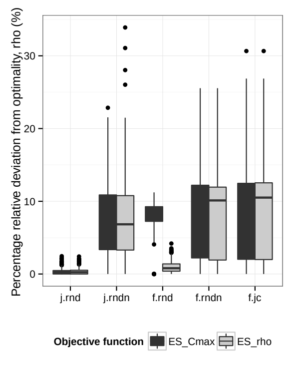
a widely-used evolutionary algorithm, Covariance Matrix Adaptation Evolution Strategy (CMA-ES), is applied directly on the desired objective function. For this study both, *a*) expected relative error, $\mathbb{E}[\rho]$; and *b*) final makespan, $\mathbb{E}[C_{\max}]$, will be considered.

Benefit No need to collect training data beforehand.

Drawback Computationally expensive to evaluate $\mathbb{E}[\cdot]$

Experiments (1)

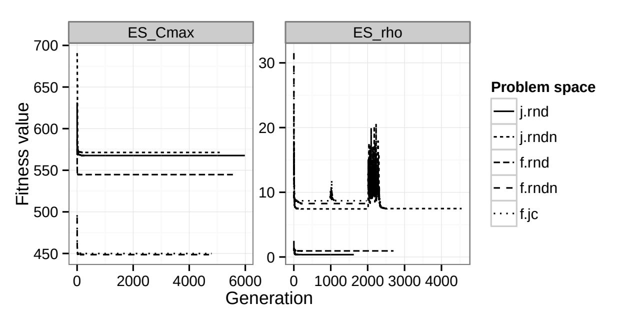
Fitness for optimising with CMA-ES



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

Experiments (2)

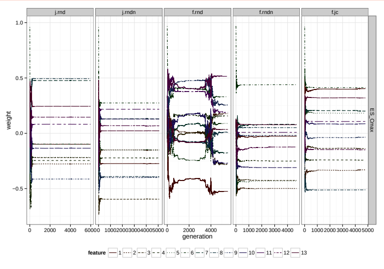
Fitness for optimising with CMA-ES

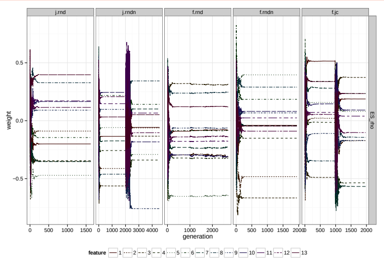


Fitness for minimising w.r.t. $C_{\max}$ and ρ, per generation of CMA-ES.

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. $C_{\max}$ (left) and ρ (right).

Summary and conclusions (1)

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

The approaches find linear weights by direct optimisation with CMA-ES

The methods significantly outperforms SDRs from the literature, and our previous work which was based on preference learning.

Summary and conclusions (2)

CMA-ES optimisation

Benefits:

- Does not rely on optimal solutions – although can help
- Scalable – model based on 6×5 successfully applied to 10×10.

Drawbacks:

- Computationally expensive .
- Limited to linear preference function $h(\cdot)$

Future Work:

- Mediate evolutionary search by use of surrogate models which indirectly estimate mean expected error w.r.t. current population without a loss in performance

Thank you for your attention

Questions?

Contact: Helga Ingimundardóttir, hei2@hi.is