Generating Training Data for Learning Linear Composite Dispatching Rules for Scheduling

Helga Ingimundardóttir
Thomas Philip Runarsson

University of Iceland

January, 2015

Outline

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.

HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRUVÍSINDASVIÐ

Case Study: JSP

Abstract

- Framework for creating dispatching rules for JSP.
- Linear classification to identify good dispatches from worse ones.
- Generate training data both from optimal and suboptimal solutions, by exploring various trajectories within the state-space.
- Sample training data using different ranking schemes.

Keywords: Scheduling • Composite dispatching rules • JSP • Generating Training Data • Trajectory Sampling Strategies • Ranking Schemes

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, $C_{\max}$.

Job Shop Scheduling (2)

Problem space distributions used in experimental studies

	name	size	N_{train}	N_{test}	note
JSP	$\mathcal{P}_{j.rnd}^{6\times5}$	6×5	500	500	random
	$\mathcal{P}_{j.rndn}^{6\times5}$	6×5	500	500	random-narrow

Job Shop Scheduling (3)

Dispatching rules (DR) for constructing JSP

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

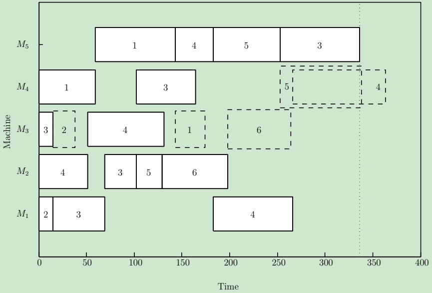
Job Shop Scheduling (4)

Features for JSP

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 (5)

Example

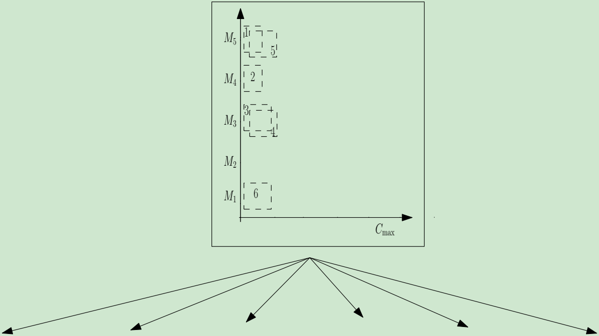


A schedule being built at step $k = 16$. The dashed boxes represent five different possible jobs that could be scheduled next using a DR.

Game-tree representation (1)

Example

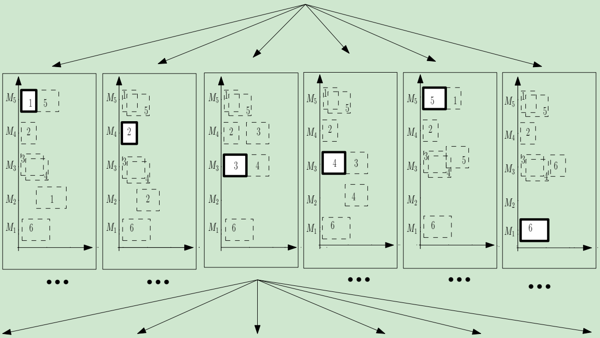
First layer (i.e. root) – empty schedule at step $k = 1$



Game-tree representation (2)

Example

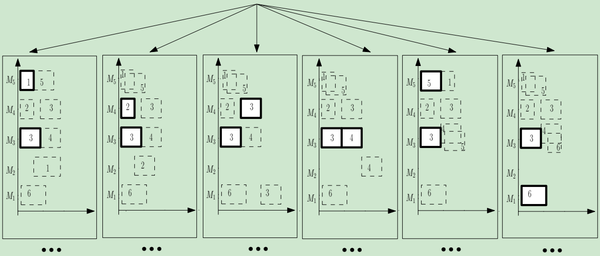
Second layer – all possible first dispatches at step $k = 2$



Game-tree representation (3)

Example

Third layer – given J_3 is dispatched first on M_3 at step $k = 3$



HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRUVÍSINDASVIÐ

Ordinal Regression (1)

Preference learning problem

Specified by a set of preference pairs:

$$S = \left\{ \{\mathbf{z}_o, +1)\}_{k=1}^{\ell}, \{\mathbf{z}_s, -1)\}_{k=1}^{\ell} \mid \forall o \in \mathcal{O}^{(k)}, s \in \mathcal{S}^{(k)} \right\} \subset \Phi \times Y$$

where the set of point/rank pairs are:

- Optimal decision: $\mathbf{z_o} = \phi^{(o)} - \phi^{(s)}$, ranked $+1$
- Suboptimal decision: $\mathbf{z_s} = \phi^{(s)} - \phi^{(o)}$, ranked -1

and $\phi_o, \phi_s \in \Phi \subset \mathcal{F}$ are features from the collected training set Φ.

Ordinal Regression (2)

- Mapping of points to ranks: $\{h(\cdot) : \Phi \mapsto Y\}$ where

$$\phi_o \succ \phi_s \quad \Leftrightarrow \quad h(\phi_o) > h(\phi_s)$$

- The preference is defined by a linear function, i.e. PREF model:

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

- Logistic regression learns the optimal parameters **w** by solving:

$$\min_{\mathbf{w}} \quad \frac{1}{2}\langle w \cdot w \rangle + C \sum_{j=1}^{|S|} \log\left(1 + e^{-y_j \langle w \cdot z_j \rangle}\right)$$

Generating preference set S (1)

A seperate DR for each dispatch iteration

- At each dispatch k a number of data pairs are created
  - for each of the N_{train} problem instance created.
- Deliberately create a separate data set for each dispatch
  - Resulting in ℓ linear scheduling rules for solving a $n \times m$ JSP.

Defining the size of the training set as $l' = |\Phi|$, gives the size of the preference set as $|S| = l = 2l'$.

- If l is too large, than sampling needs to be done.

Generating preference set S (2)

Previous sampling approach

The strategy was to follow some single optimal job $j \in \mathcal{O}^{(k)}$, thus creating $|\mathcal{O}^{(k)}| \cdot |\mathcal{S}^{(k)}|$ feature pairs at each dispatch k, resulting in a training size of:

$$l' = \sum_{q=1}^{N_{\text{train}}} \left(\sum_{k=1}^{\ell} |\mathcal{O}^{(k)}| \cdot |\mathcal{S}^{(k)}| \right)$$

For the data distribution considered there, this simple sampling was sufficient for a favourable outcome. However for a considerably harder data distribution this strategy did not work well.

Generating preference set S (3)

Trajectory sampling strategies explored for adding features to Φ

- Φ^{opt} follow some (random) optimal task
- Φ^{cma} follow the task corresponding to highest priority, computed with fixed weights $\mathbf{w}$, which were obtained by optimising with CMA-ES.
- Φ^{mwr} follow the SDR most work remaining (MWR).
- Φ^{rnd} follow some random task.
- Φ^{all} union of all of the above, i.e.,

$$\Phi^{all} = \Phi^{opt} \cup \Phi^{cma} \cup \Phi^{mwr} \cup \Phi^{rnd}$$

Generating preference set S (4)

Ranking schemes implemented for adding preference pairs to S

- S_b all opt rankings r_1 vs. all possible subopt rankings r_i, $i \in \{2, ..., n'\}$
- S_f full subsequent rankings, i.e., all combinations of r_i and r_{i+1} for all $i \in \{1, ..., n'\}$.
- S_p partial subsequent rankings, similar ot S_f except if there are more than one operation with the same ranking, only one is needed to be compared to subsequent rank, i.e., $S_p \subset S_f$.
- S_a union of all of the above, i.e.,

$$S_a = S_b \cup S_f \cup S_p$$

where $r_1 > r_2 > \cdots > r_{n'}$ $(n' \leq n)$ are the rankings of $\mathcal{R}^{(k)}$.

Evolutionary search

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

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

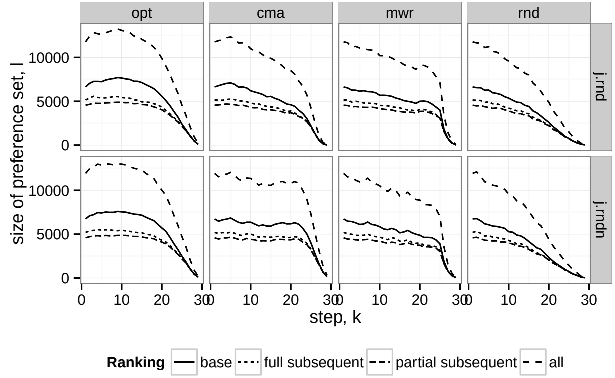
a widely-used evolutionary algorithm, Covariance Matrix Adaptation Evolution Strategy (CMA-ES), is applied directly on the objective function.

Benefit No need to collect training data beforehand.

Drawback Computationally expensive to evaluate $\mathbb{E}\left[C_{\max}\right]$

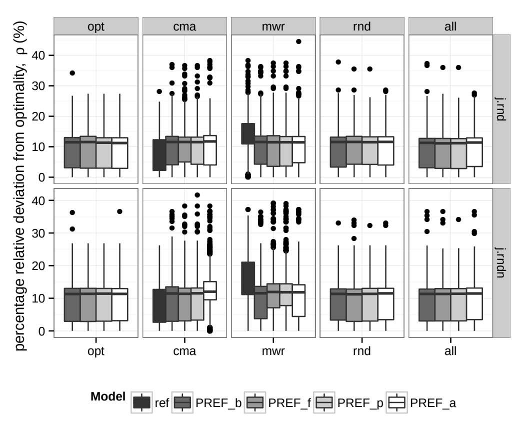
Experiments (1)

Size of preference set, $l = |S|$



Experiments (2)

Box-plot for PREF models using test set



Experiments (3)

Trajectory sampling strategies

- Learning preferences from good scheduling rules can give favourable results, e.g. Φ^{cma} and Φ^{mwr}.
- Tracking only optimal paths (Φ^{opt}) yield a generally lower mean relative error – although there was no statistical difference with Φ^{rnd}, implying the model has diverged from the learned optimal features set and is inept to determine good dispatches from that point onward.
- For $\mathcal{P}^{6\times5}_{j.rnd}$ the best model was based on Φ^{all}, where the suboptimal trajectories aid Φ^{opt} by adding a greater variety of preference pairs for getting out of local minima.

Experiments (4)

Results for ranking schemes

- No statistical difference between ranking schemes. However, opting for a smaller preference set then S_p is preferred.

Summary and conclusions (1)

- Introduced a framework for learning linear composite dispatch rules for scheduling.
- The approach is based on preference learning and its success is highly dependent on the preference pairs introduced to the system, i.e., the trajectories explored through the state-space. Here, optimal and suboptimal trajectories are of value.
- By partial subsequent ranking scheme it's possible to reduce the preference set without loss of performance.

Summary and conclusions (2)

Future work

- Instead of creating preference set based on collecting trajectories based on arbitrary dispatching rules (or models), it's worthwhile to continue with $PREF^{opt}$ model and collect preference set following its learned policy and use that to create a new model similar to Φ^{all}.
- Namely use the model to update itself – also known as imitation learning.
- Preliminary experiments have shown that a $PREF^{opt}$ with 14.07% mean relative error can be improved to 8.52% with only one unsupervised iteration, or 9.98% if iteration is supervised.[1]

[1] Supervised set-up: 50% chance $PREF^{opt}$ used, 50% optimal move chosen.

Thank you for your attention

HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRUVÍSINDASVIÐ

Questions?

Contact: Helga Ingimundardóttir, hei2@hi.is