ALICE

Analysis & Learning Iterative Consecutive Executions

Helga Ingimundardóttir

University of Iceland

June 30, 2016

Introduction

Motivation:

- ⋆ The general goal is to train optimisation algorithms, for an arbitrary problem domain, using data.

Contribution:

- ⋆ The main contribution of this thesis is towards a better understanding of how this training data should be constructed.

Introduction

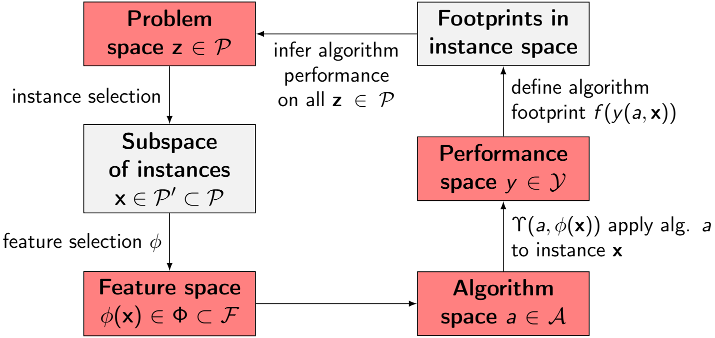
Motivation:

- ⋆ The general goal is to train optimisation algorithms, for an arbitrary problem domain, using data.

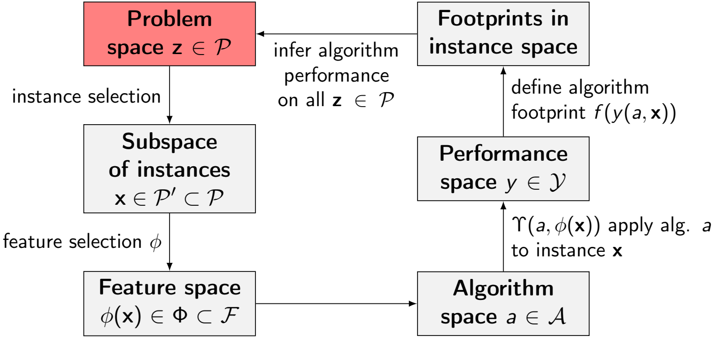
Contribution:

- ⋆ The main contribution of this thesis is towards a better understanding of how this training data should be constructed.

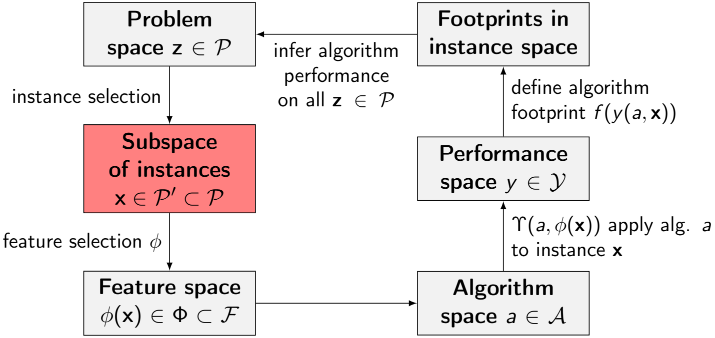
Framework for Algorithm Learning



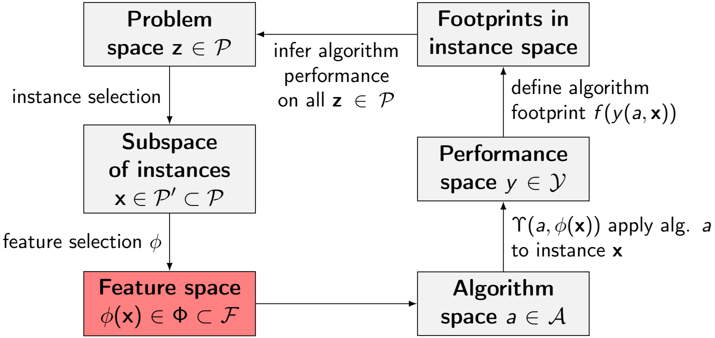
Framework for Algorithm Learning



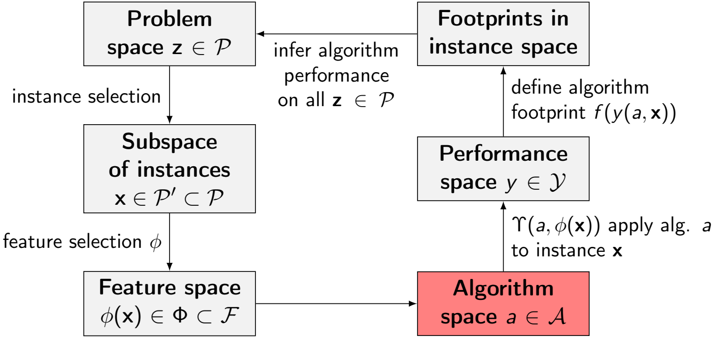
Framework for Algorithm Learning



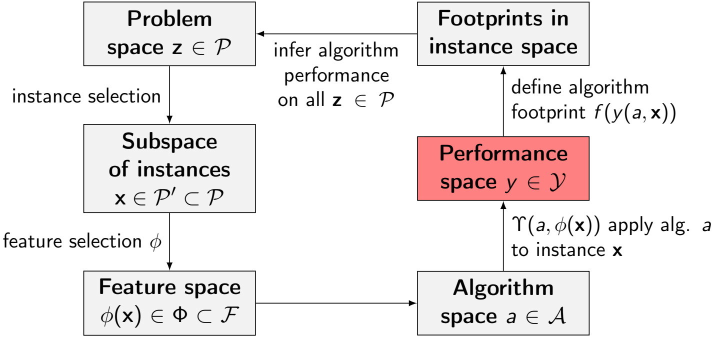
Framework for Algorithm Learning



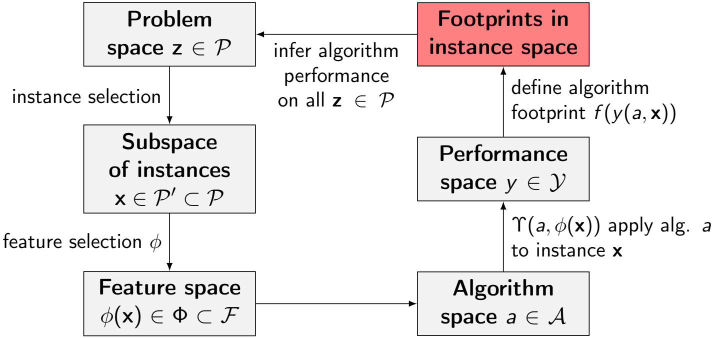
Framework for Algorithm Learning



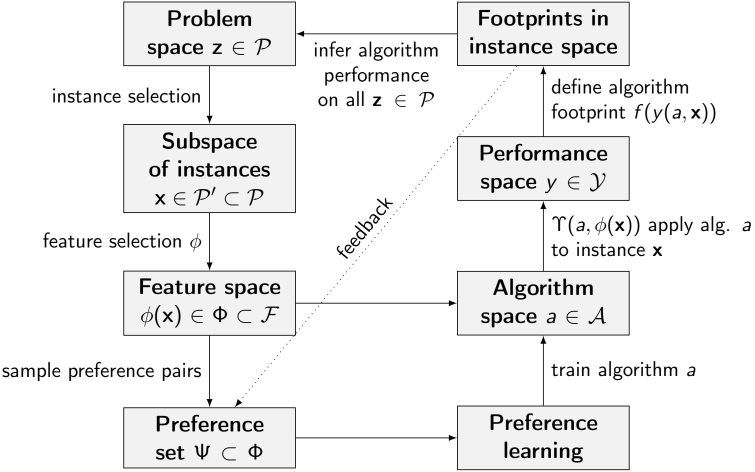
Framework for Algorithm Learning



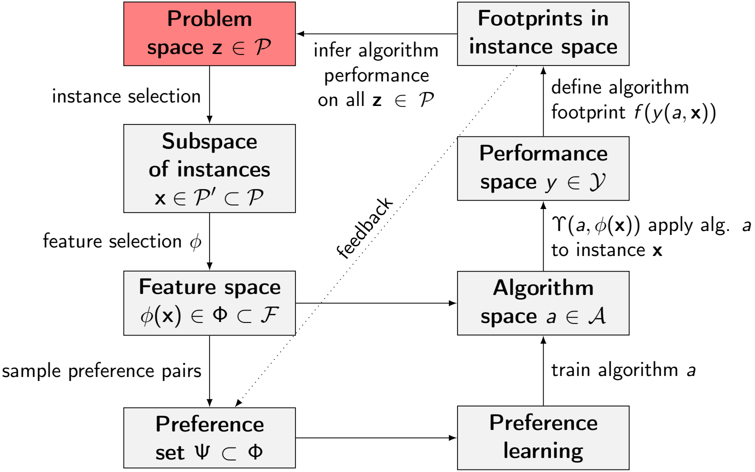
Framework for Algorithm Learning



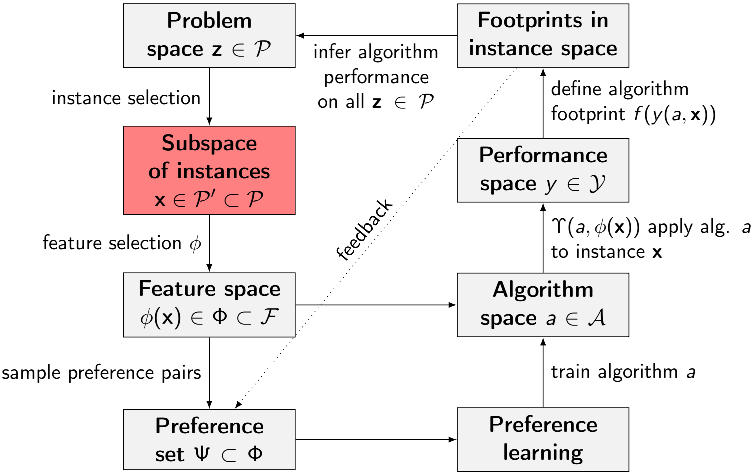
Framework for Algorithm Learning



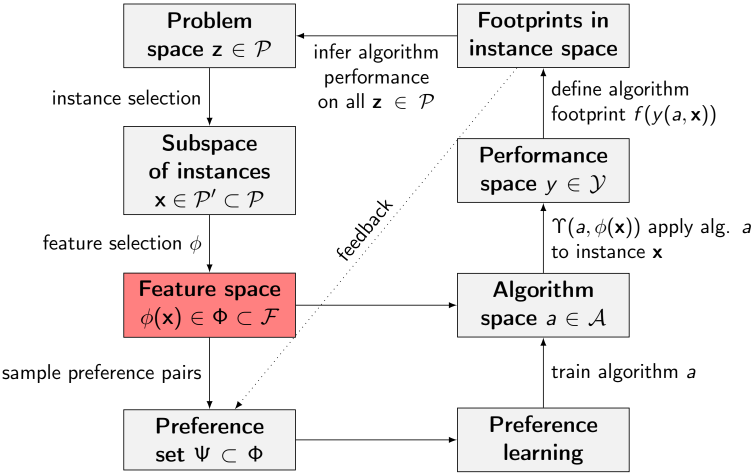
Framework for Algorithm Learning



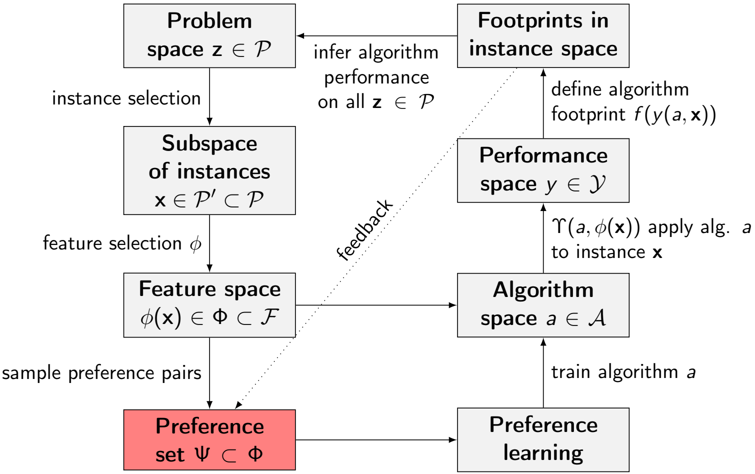
Framework for Algorithm Learning



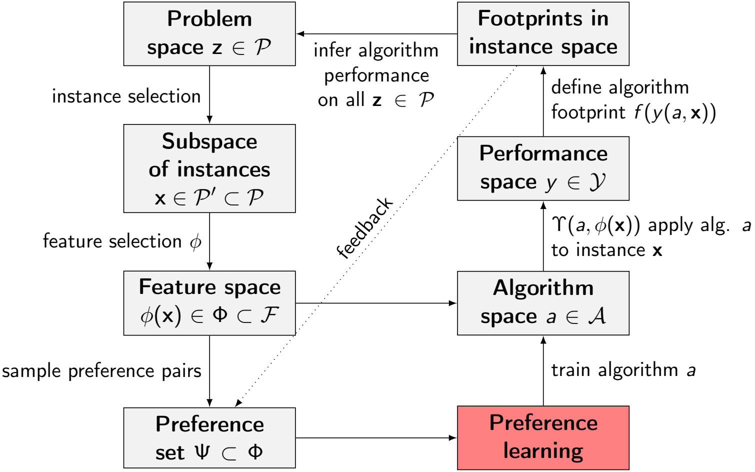
Framework for Algorithm Learning



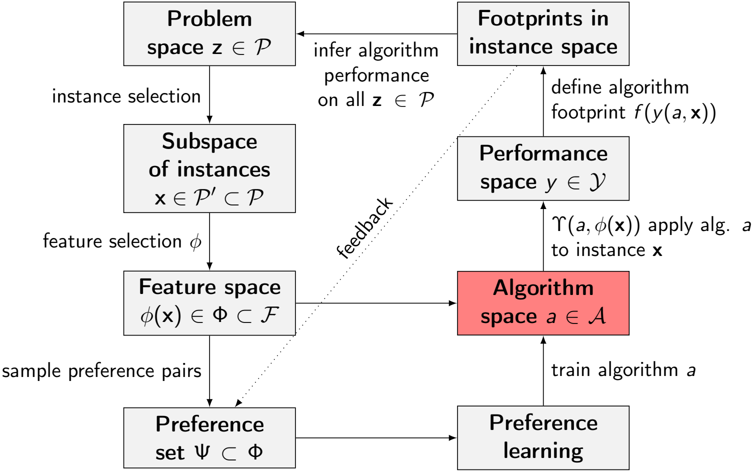
Framework for Algorithm Learning



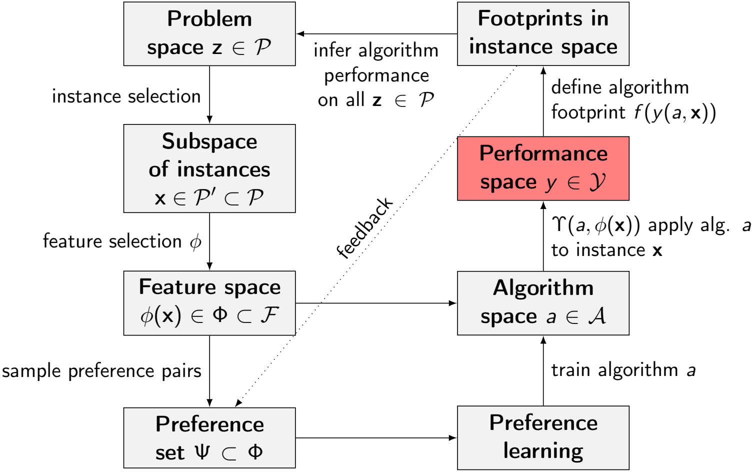
Framework for Algorithm Learning



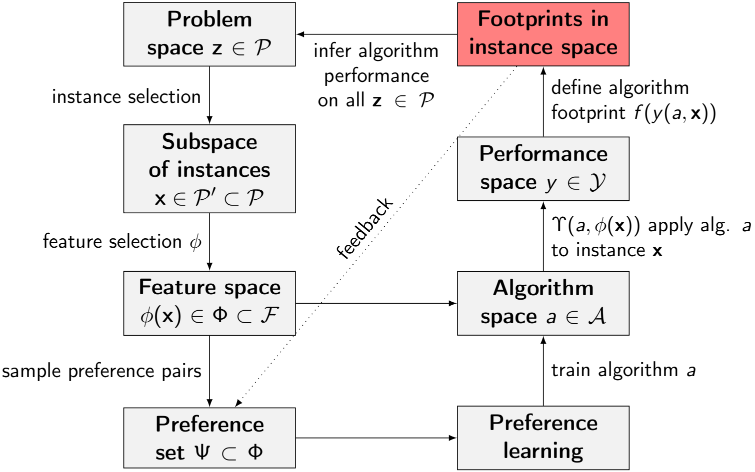
Framework for Algorithm Learning



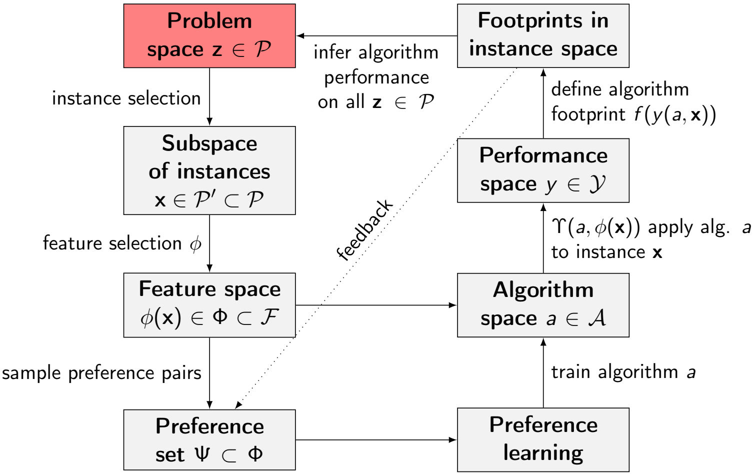
Framework for Algorithm Learning



Framework for Algorithm Learning



Framework for Algorithm Learning



Mad Hatter tea-party (definition)

Mad Hatter tea-party (definition)

The attending guests:

J_1) Alice
J_2) March Hare
J_3) Dormouse
J_4) Mad Hatter.

They all have to:

M_1) have wine or pour tea
M_2) spread butter
M_3) get a haircut
M_4) check the time of the broken watch
M_5) say what they mean.

This can be considered as is a typical 4×5 job-shop, where:

- ⋆ our guests are the jobs
- ⋆ their tasks are the machines
- ⋆ objective is to minimise $C_{\max}$ (when Alice can leave).

Mad Hatter tea-party (definition)

The attending guests:

J_1) Alice

J_2) March Hare

J_3) Dormouse

J_4) Mad Hatter.

They all have to:

M_1) have wine or pour tea

M_2) spread butter

M_3) get a haircut

M_4) check the time of the broken watch

M_5) say what they mean.

This can be considered as is a typical 4×5 job-shop, where:

- ⋆ our guests are the jobs
- ⋆ their tasks are the machines
- ⋆ objective is to minimise $C_{\max}$ (when Alice can leave).

Mad Hatter tea-party (definition)

The attending guests:

J_1) Alice
J_2) March Hare
J_3) Dormouse
J_4) Mad Hatter.

They all have to:

M_1) have wine or pour tea
M_2) spread butter
M_3) get a haircut
M_4) check the time of the broken watch
M_5) say what they mean.

This can be considered as is a typical 4×5 job-shop, where:

- ⋆ our guests are the jobs
- ⋆ their tasks are the machines
- ⋆ objective is to minimise $C_{\max}$ (when Alice can leave).

Mad Hatter tea-party (definition)

The attending guests:

J_1) Alice

J_2) March Hare

J_3) Dormouse

J_4) Mad Hatter.

They all have to:

M_1) have wine or pour tea

M_2) spread butter

M_3) get a haircut

M_4) check the time of the broken watch

M_5) say what they mean.

This can be considered as is a typical 4×5 job-shop, where:

- ⋆ our guests are the jobs
- ⋆ their tasks are the machines
- ⋆ objective is to minimise $C_{\max}$ (when Alice can leave).

Mad Hatter tea-party (k-solutions)

Start: $k = 0$

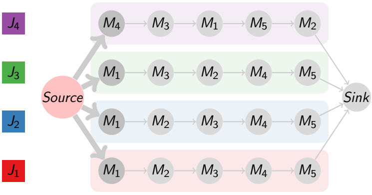


Figure: Disjunctive graph

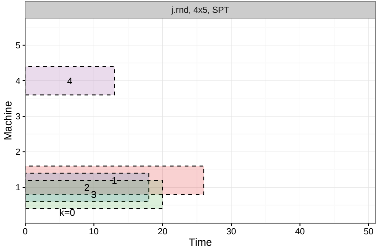


Figure: Gantt chart

ALICE
Helga
Introduction
Problem space
Subspace of instances
Feature space
Algorithm space
Performance space
Footprints in instance space
Preference set
Preference learning
Conclusions

Midway: $k = 10$

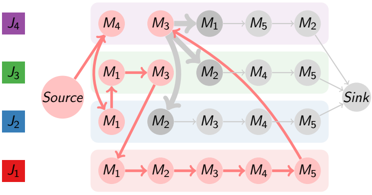


Figure: Disjunctive graph

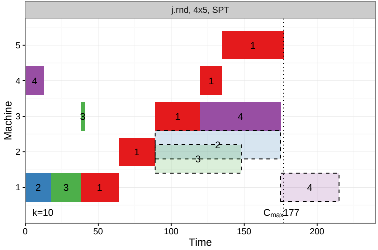


Figure: Gantt chart

Mad Hatter tea-party (k-solutions)

Finish: $k = 20$

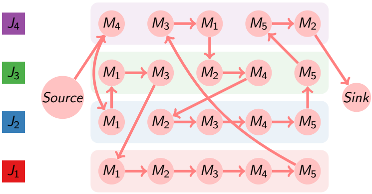


Figure: Disjunctive graph

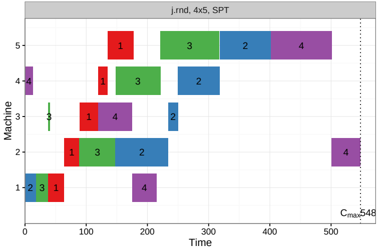
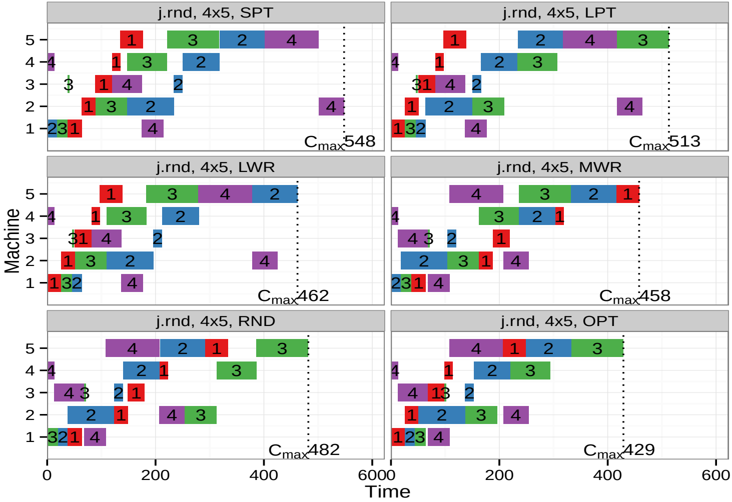
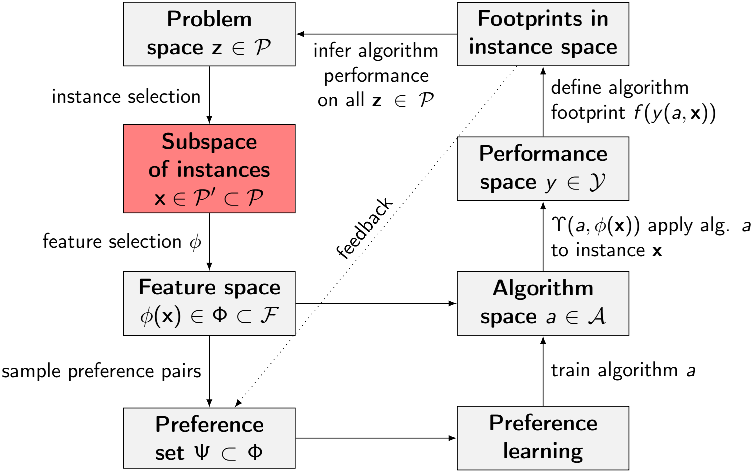


Figure: Gantt chart

Mad Hatter tea-party (K-solutions)



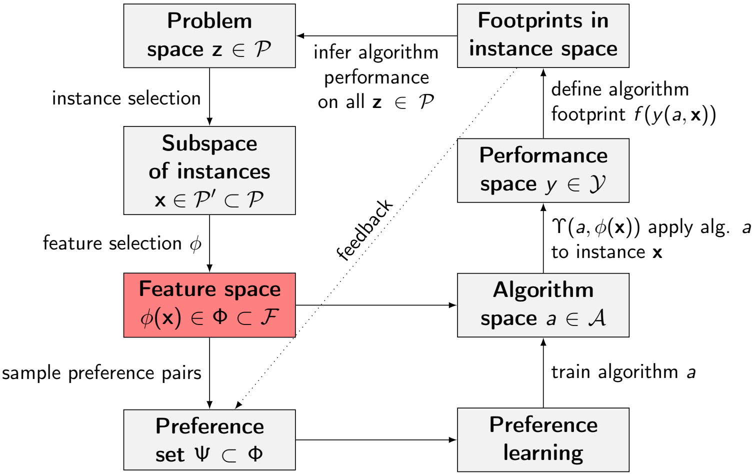
Framework for Algorithm Learning



Problem instance generators

	name	size ($n \times m$)	N_{train}	N_{test}	note
JSP	$\mathcal{P}^{6\times5}_{j.rnd}$	6×5	500	500	random
	$\mathcal{P}^{6\times5}_{j.rndn}$	6×5	500	500	random-narrow
	$\mathcal{P}^{6\times5}_{j.rnd,J_1}$	6×5	500	500	random with job variation
	$\mathcal{P}^{6\times5}_{j.rnd,M_1}$	6×5	500	500	random with machine variation
	$\mathcal{P}^{10\times10}_{j.rnd}$	10×10	300	200	random
	$\mathcal{P}^{10\times10}_{j.rndn}$	10×10	300	200	random-narrow
	$\mathcal{P}^{10\times10}_{j.rnd,J_1}$	10×10	300	200	random with job variation
	$\mathcal{P}^{10\times10}_{j.rnd,M_1}$	10×10	300	200	random with machine variation
	$\mathcal{P}_{JSP.ORLIB}$	various	–	82	various
FSP	$\mathcal{P}^{6\times5}_{f.rnd}$	6×5	500	500	random
	$\mathcal{P}^{6\times5}_{f.rndn}$	6×5	500	500	random-narrow
	$\mathcal{P}^{6\times5}_{f.jc}$	6×5	500	500	job-correlated
	$\mathcal{P}^{6\times5}_{f.mc}$	6×5	500	500	machine-correlated
	$\mathcal{P}^{6\times5}_{f.mxc}$	6×5	500	500	mixed-correlation
	$\mathcal{P}^{10\times10}_{f.rnd}$	10×10	300	200	random
	$\mathcal{P}_{FPS.ORLIB}$	various	–	31	various

Framework for Algorithm Learning



Features for JSP

job	ϕ_1	job processing time
	ϕ_2	job start-time
	ϕ_3	job end-time
	ϕ_4	job arrival time
	ϕ_5	time job had to wait
	ϕ_6	total processing time for job
	ϕ_7	total work remaining for job
	ϕ_8	number of assigned operations for job
machine	ϕ_9	when machine is next free
	ϕ_{10}	total processing time for machine
	ϕ_{11}	total work remaining for machine
	ϕ_{12}	number of assigned operations for machine
	ϕ_{13}	change in idle time by assignment
	ϕ_{14}	total idle time for machine
	ϕ_{15}	total idle time for all machines
	ϕ_{16}	current makespan
final makespan	ϕ_{17}	final makespan using SPT
	ϕ_{18}	final makespan using LPT
	ϕ_{19}	final makespan using LWR
	ϕ_{20}	final makespan using MWR
	ϕ_{RND}	final makespans using 100 random rollouts
	ϕ_{21}	mean for ϕ_{RND}
	ϕ_{22}	standard deviation for ϕ_{RND}
	ϕ_{23}	minimum value for ϕ_{RND}
	ϕ_{24}	maximum value for ϕ_{RND}

Features for JSP

job	ϕ_1	job processing time
	ϕ_2	job start-time
	ϕ_3	job end-time
	ϕ_4	job arrival time
	ϕ_5	time job had to wait
	ϕ_6	total processing time for job
	ϕ_7	total work remaining for job
	ϕ_8	number of assigned operations for job
machine	ϕ_9	when machine is next free
	ϕ_{10}	total processing time for machine
	ϕ_{11}	total work remaining for machine
	ϕ_{12}	number of assigned operations for machine
	ϕ_{13}	change in idle time by assignment
	ϕ_{14}	total idle time for machine
	ϕ_{15}	total idle time for all machines
	ϕ_{16}	current makespan
final makespan	ϕ_{17}	final makespan using SPT
	ϕ_{18}	final makespan using LPT
	ϕ_{19}	final makespan using LWR
	ϕ_{20}	final makespan using MWR
	ϕ_{RND}	final makespans using 100 random rollouts
	ϕ_{21}	mean for ϕ_{RND}
	ϕ_{22}	standard deviation for ϕ_{RND}
	ϕ_{23}	minimum value for ϕ_{RND}
	ϕ_{24}	maximum value for ϕ_{RND}

Features for JSP

job	ϕ_1	job processing time
	ϕ_2	job start-time
	ϕ_3	job end-time
	ϕ_4	job arrival time
	ϕ_5	time job had to wait
	ϕ_6	total processing time for job
	ϕ_7	total work remaining for job
	ϕ_8	number of assigned operations for job
machine	ϕ_9	when machine is next free
	ϕ_{10}	total processing time for machine
	ϕ_{11}	total work remaining for machine
	ϕ_{12}	number of assigned operations for machine
	ϕ_{13}	change in idle time by assignment
	ϕ_{14}	total idle time for machine
	ϕ_{15}	total idle time for all machines
	ϕ_{16}	current makespan
final makespan	ϕ_{17}	final makespan using SPT
	ϕ_{18}	final makespan using LPT
	ϕ_{19}	final makespan using LWR
	ϕ_{20}	final makespan using MWR
	ϕ_{RND}	final makespans using 100 random rollouts
	ϕ_{21}	mean for ϕ_{RND}
	ϕ_{22}	standard deviation for ϕ_{RND}
	ϕ_{23}	minimum value for ϕ_{RND}
	ϕ_{24}	maximum value for ϕ_{RND}

Trajectory strategies for Φ

Following the policy:

- $\star$ (Φ^{OPT}) expert $\pi_\star$.
- $\star$ (Φ^{SPT}) shortest processing time (SPT).
- $\star$ (Φ^{LPT}) longest processing time (LPT).
- $\star$ (Φ^{LWR}) least work remaining (LWR).
- $\star$ (Φ^{MWR}) most work remaining (MWR).
- $\star$ (Φ^{RND}) random policy (RND).
- $\star$ ($\Phi^{\text{ES}.\rho}$) the policy obtained by optimising with CMA-ES.
- $\star$ (Φ^{ALL}) union of all of the above.

Trajectory strategies for Φ

Following the policy:

- (Φ^{OPT}) expert $\pi_\star$.
- (Φ^{SPT}) shortest processing time (SPT).
- (Φ^{LPT}) longest processing time (LPT).
- (Φ^{LWR}) least work remaining (LWR).
- (Φ^{MWR}) most work remaining (MWR).
- (Φ^{RND}) random policy (RND).
- ($\Phi^{\mathrm{ES}.\rho}$) the policy obtained by optimising with CMA-ES.
- (Φ^{ALL}) union of all of the above.

Trajectory strategies for Φ

Following the policy:

- $\star$ (Φ^{OPT}) expert $\pi_\star$.
- $\star$ (Φ^{SPT}) shortest processing time (SPT).
- $\star$ (Φ^{LPT}) longest processing time (LPT).
- $\star$ (Φ^{LWR}) least work remaining (LWR).
- $\star$ (Φ^{MWR}) most work remaining (MWR).
- $\star$ (Φ^{RND}) random policy (RND).
- $\star$ ($\Phi^{\mathrm{ES}.\rho}$) the policy obtained by optimising with CMA-ES.
- $\star$ (Φ^{ALL}) union of all of the above.

Trajectory strategies for Φ

Following the policy:

- (Φ^{OPT}) expert $\pi_\star$.
- (Φ^{SPT}) shortest processing time (SPT).
- (Φ^{LPT}) longest processing time (LPT).
- (Φ^{LWR}) least work remaining (LWR).
- (Φ^{MWR}) most work remaining (MWR).
- (Φ^{RND}) random policy (RND).
- ($\Phi^{\mathrm{ES}.\rho}$) the policy obtained by optimising with CMA-ES.
- (Φ^{ALL}) union of all of the above.

Trajectory strategies for Φ

Following the policy:

- $\star$ (Φ^{OPT}) expert $\pi_\star$.
- $\star$ (Φ^{SPT}) shortest processing time (SPT).
- $\star$ (Φ^{LPT}) longest processing time (LPT).
- $\star$ (Φ^{LWR}) least work remaining (LWR).
- $\star$ (Φ^{MWR}) most work remaining (MWR).
- $\star$ (Φ^{RND}) random policy (RND).
- $\star$ ($\Phi^{\mathrm{ES}.\rho}$) the policy obtained by optimising with CMA-ES.
- $\star$ (Φ^{ALL}) union of all of the above.

Trajectory strategies for Φ

Following the policy:

- $\star$ (Φ^{OPT}) expert $\pi_\star$.
- $\star$ (Φ^{SPT}) shortest processing time (SPT).
- $\star$ (Φ^{LPT}) longest processing time (LPT).
- $\star$ (Φ^{LWR}) least work remaining (LWR).
- $\star$ (Φ^{MWR}) most work remaining (MWR).
- $\star$ (Φ^{RND}) random policy (RND).
- $\star$ ($\Phi^{\mathrm{ES}.\rho}$) the policy obtained by optimising with CMA-ES.
- $\star$ (Φ^{ALL}) union of all of the above.

Trajectory strategies for Φ

ALICE
Helga
Introduction
Problem space
Subspace of instances
Feature space
Algorithm space
Performance space
Footprints in instance space
Preference set
Preference learning
Conclusions

Following the policy:

- (Φ^{OPT}) expert $\pi_\star$.
- (Φ^{SPT}) shortest processing time (SPT).
- (Φ^{LPT}) longest processing time (LPT).
- (Φ^{LWR}) least work remaining (LWR).
- (Φ^{MWR}) most work remaining (MWR).
- (Φ^{RND}) random policy (RND).
- ($\Phi^{\text{ES}.\rho}$) the policy obtained by optimising with CMA-ES.
- (Φ^{ALL}) union of all of the above.

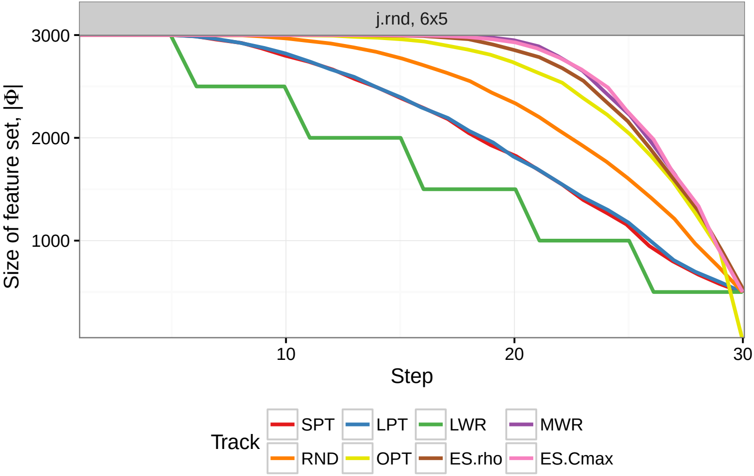
Trajectory strategies for Φ

Following the policy:

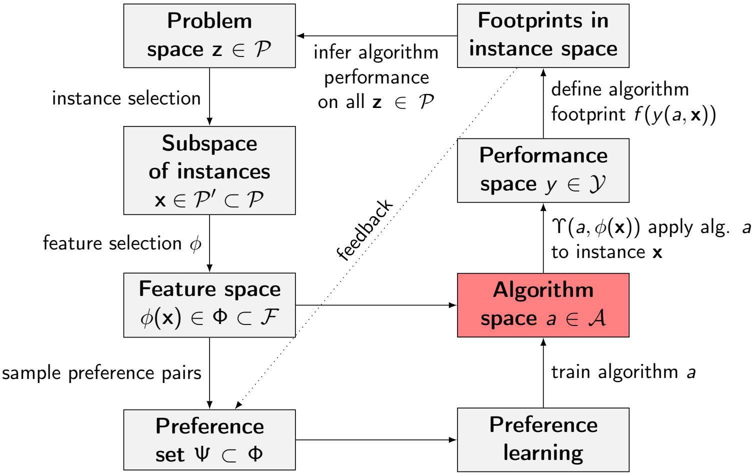
- $\star$ (Φ^{OPT}) expert $\pi_\star$.
- $\star$ (Φ^{SPT}) shortest processing time (SPT).
- $\star$ (Φ^{LPT}) longest processing time (LPT).
- $\star$ (Φ^{LWR}) least work remaining (LWR).
- $\star$ (Φ^{MWR}) most work remaining (MWR).
- $\star$ (Φ^{RND}) random policy (RND).
- $\star$ ($\Phi^{\mathrm{ES}.\rho}$) the policy obtained by optimising with CMA-ES.
- $\star$ (Φ^{ALL}) union of all of the above.

Sampled size of $|\Phi(k)|$

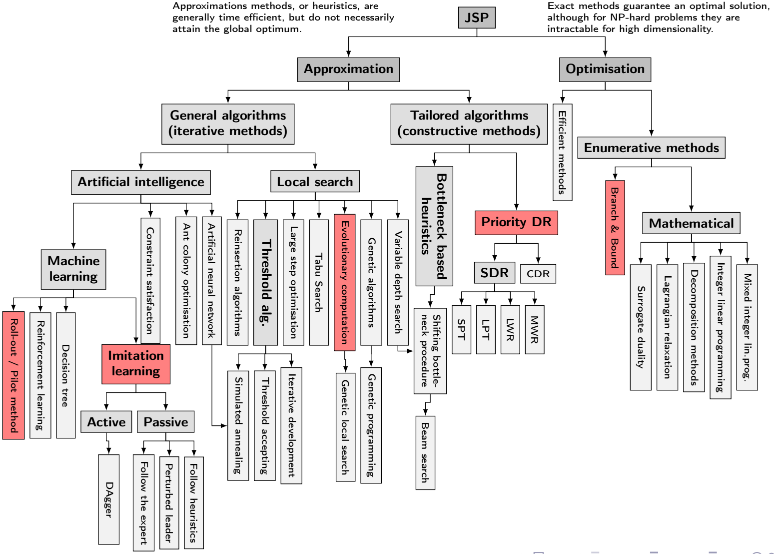
$(6 \times 5, N_{train} = 500)$



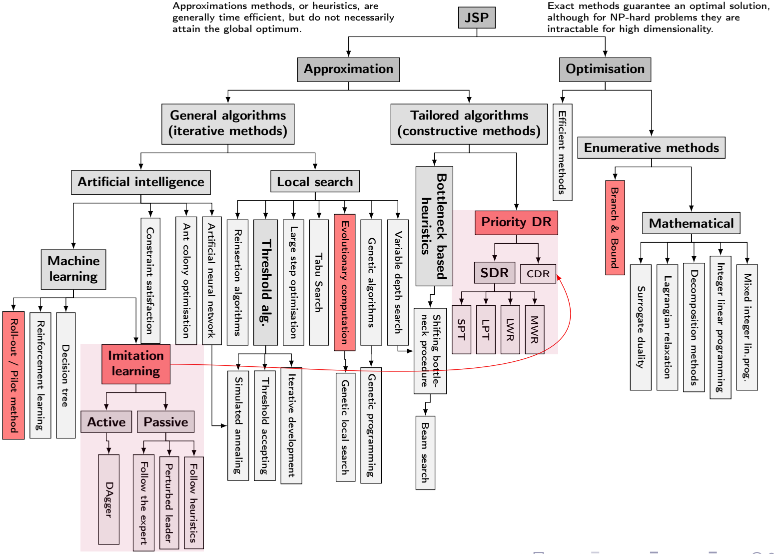
Framework for Algorithm Learning



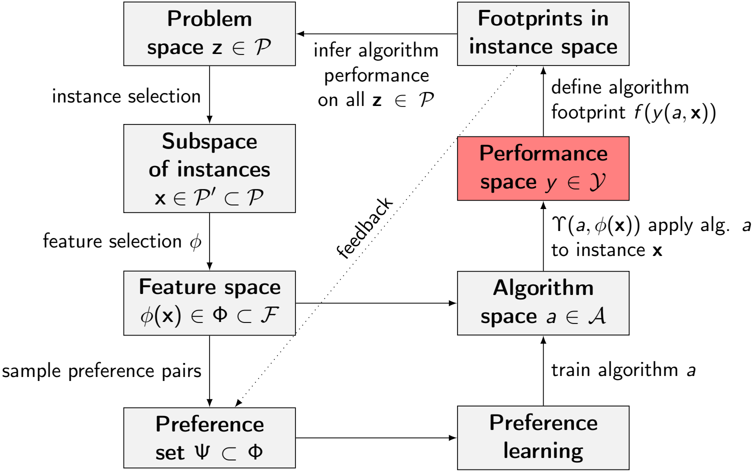
Various Methods for Solving JSP



Various Methods for Solving JSP



Framework for Algorithm Learning



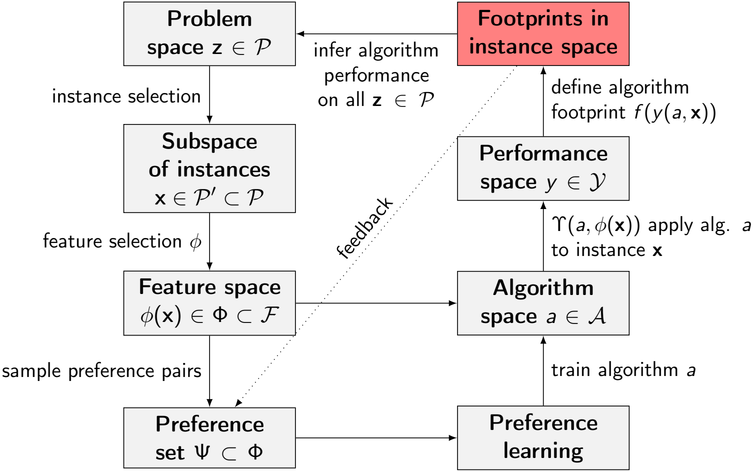
Performance measure

Performance of policy π compared with its optimal makespan, found using an expert policy, $\pi_\star$, is the following loss function:

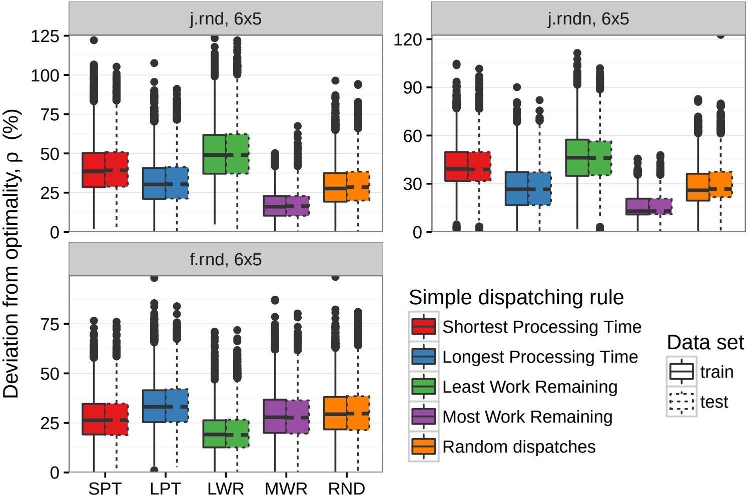
$$\rho = \frac{C_{\text{max}}^{\pi} - C_{\text{max}}^{\pi_\star}}{C_{\text{max}}^{\pi_\star}} \cdot 100\%$$

The goal is to minimise this discrepancy between predicted value and true outcome.

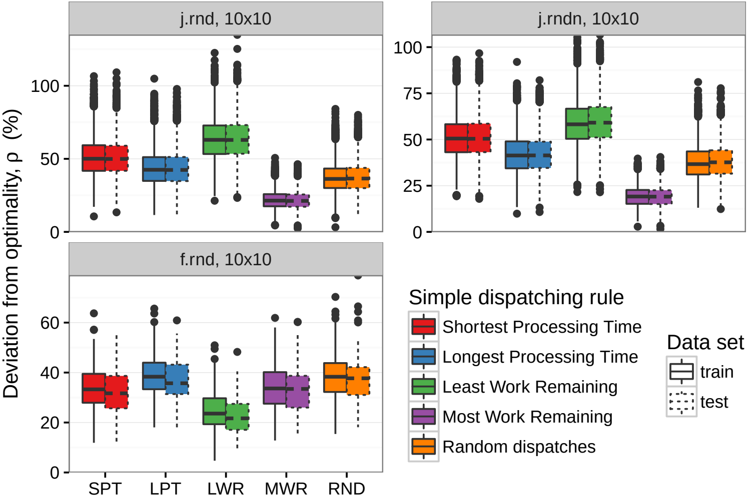
Framework for Algorithm Learning



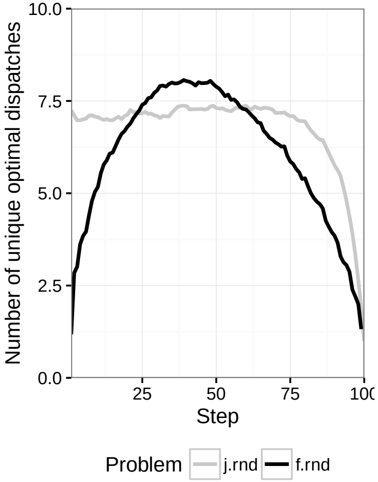
Deviation from optimality, ρ



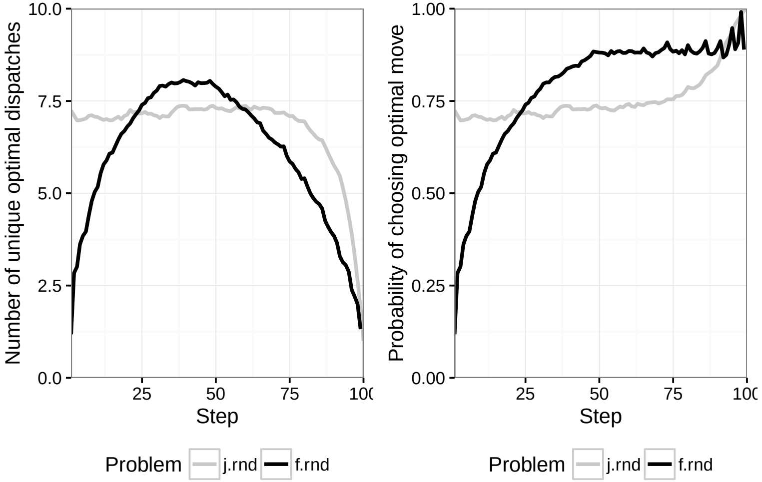
Deviation from optimality, ρ



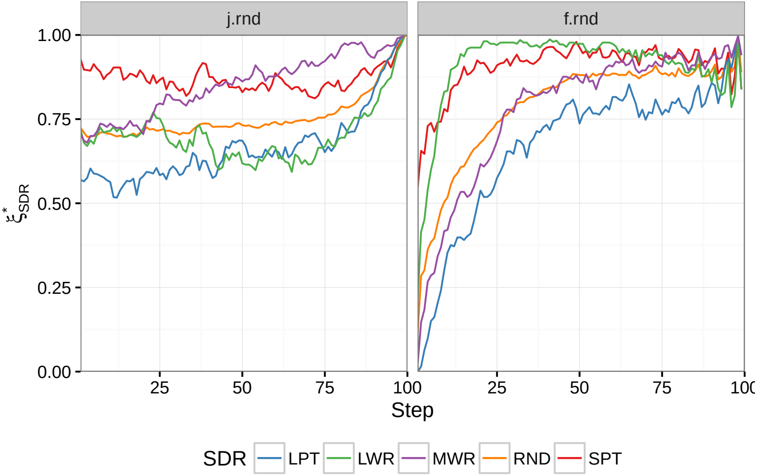
Making optimal decisions, $\xi_\pi^\star$



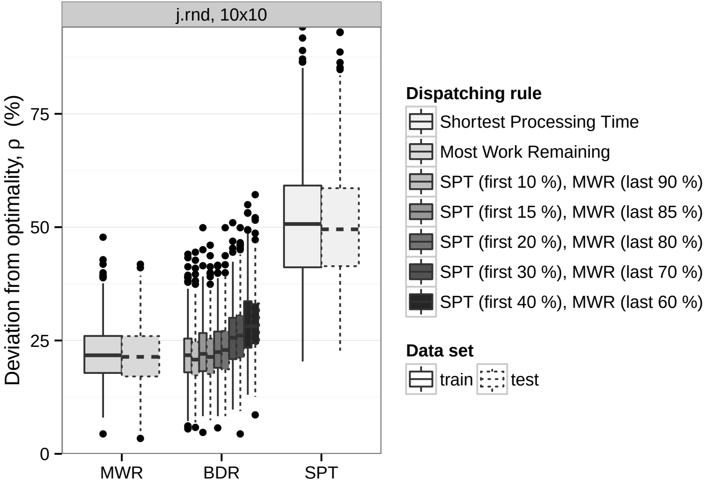
Making optimal decisions, $\xi^\star_\pi$



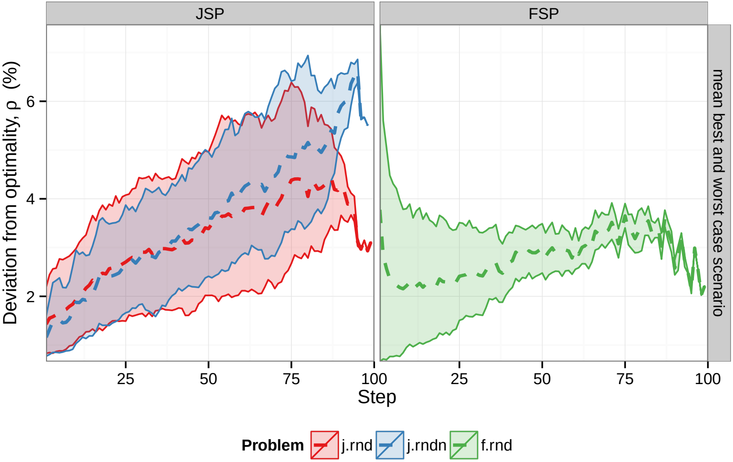
Probability of SDR being optimal, $\xi^{\star}_{\langle \text{SDR} \rangle}$



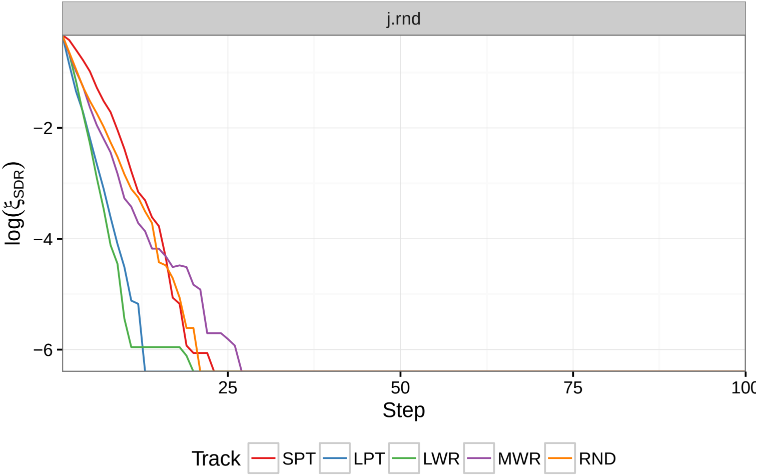
Blended dispatching rule



Impact of suboptimal decision, $\{\zeta^{\star}_{\min}, \zeta^{\star}_{\max}\}$

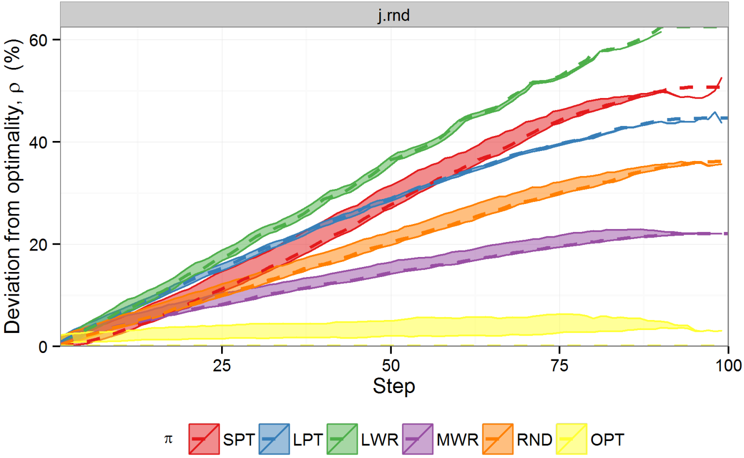


Probability of SDR being optimal, $\xi_{\langle \mathrm{SDR} \rangle}$

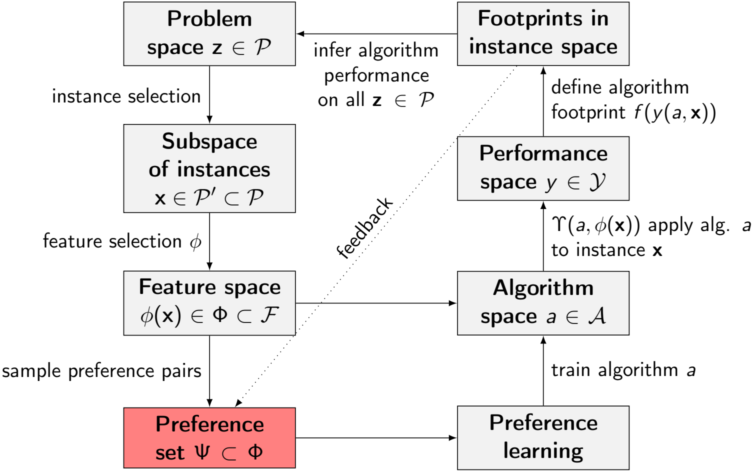


Impact of suboptimal decision, $\{\zeta^{\pi}_{\min}, \zeta^{\pi}_{\max}\}$

ALICE
Helga
Introduction
Problem space
Subspace of instances
Feature space
Algorithm space
Performance space
Footprints in instance space
Preference set
Preference learning
Conclusions



Framework for Algorithm Learning



Generating training data

ALICE framework for creating dispatching rules:

- Linear classification to identify good dispatches, from worse ones.
- Generate feature set, $\Phi \subset \mathcal{F}$, both from
  - optimal solutions, ϕ^o
  - suboptimal solutions, ϕ^s

  by exploring various trajectories within the feature-space (where $\phi^o, \phi^s \in \mathcal{F}$).
- Sample Φ to create training set Ψ with rank pairs:
  - optimal decision, $(\mathbf{z}^o, y_o) = (\phi^o - \phi^s, +1)$
  - suboptimal decision, $(\mathbf{z}^s, y_s) = (\phi^s - \phi^o, -1)$

  using different ranking schemes (where $\mathbf{z}^o, \mathbf{z}^s \in \Psi$)
- Sample Ψ using stepwise bias for time independent policy.

Generating training data

ALICE framework for creating dispatching rules:

- Linear classification to identify good dispatches, from worse ones.
- Generate feature set, $\Phi \subset \mathcal{F}$, both from
  - optimal solutions, ϕ^o
  - suboptimal solutions, ϕ^s

  by exploring various trajectories within the feature-space (where $\phi^o, \phi^s \in \mathcal{F}$).
- Sample Φ to create training set Ψ with rank pairs:
  - optimal decision, $(\mathbf{z}^o, y_o) = (\phi^o - \phi^s, +1)$
  - suboptimal decision, $(\mathbf{z}^s, y_s) = (\phi^s - \phi^o, -1)$

  using different ranking schemes (where $\mathbf{z}^o, \mathbf{z}^s \in \Psi$)
- Sample Ψ using stepwise bias for time independent policy.

Generating training data

ALICE framework for creating dispatching rules:

- ⋆ Linear classification to identify good dispatches, from worse ones.
- ⋆ Generate feature set, $\Phi \subset \mathcal{F}$, both from
  - ⋆ optimal solutions, ϕ^o
  - ⋆ suboptimal solutions, ϕ^s

  by exploring various trajectories within the feature-space (where $\phi^o, \phi^s \in \mathcal{F}$).
- ⋆ Sample Φ to create training set Ψ with rank pairs:
  - ⋆ optimal decision, $(\mathbf{z}^o, y_o) = (\phi^o - \phi^s, +1)$
  - ⋆ suboptimal decision, $(\mathbf{z}^s, y_s) = (\phi^s - \phi^o, -1)$

  using different ranking schemes (where $\mathbf{z}^o, \mathbf{z}^s \in \Psi$)
- ⋆ Sample Ψ using stepwise bias for time independent policy.

Generating training data

ALICE framework for creating dispatching rules:

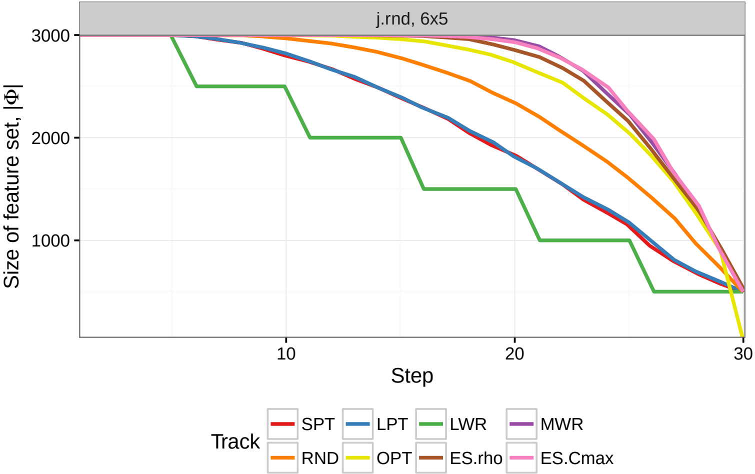
- ⋆ Linear classification to identify good dispatches, from worse ones.
- ⋆ Generate feature set, $\Phi \subset \mathcal{F}$, both from
  - ⋆ optimal solutions, ϕ^o
  - ⋆ suboptimal solutions, ϕ^s

  by exploring various trajectories within the feature-space (where $\phi^o, \phi^s \in \mathcal{F}$).
- ⋆ Sample Φ to create training set Ψ with rank pairs:
  - ⋆ optimal decision, $(\mathbf{z}^o, y_o) = (\phi^o - \phi^s, +1)$
  - ⋆ suboptimal decision, $(\mathbf{z}^s, y_s) = (\phi^s - \phi^o, -1)$

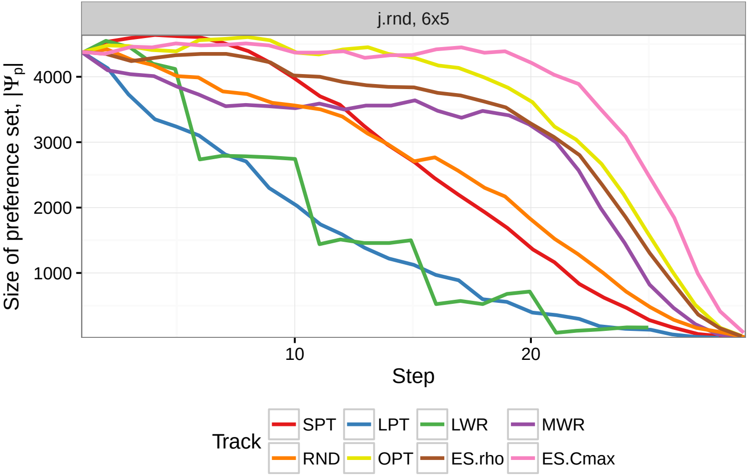
  using different ranking schemes (where $\mathbf{z}^o, \mathbf{z}^s \in \Psi$)
- ⋆ Sample Ψ using stepwise bias for time independent policy.

Sampled size of $|\Phi(k)|$

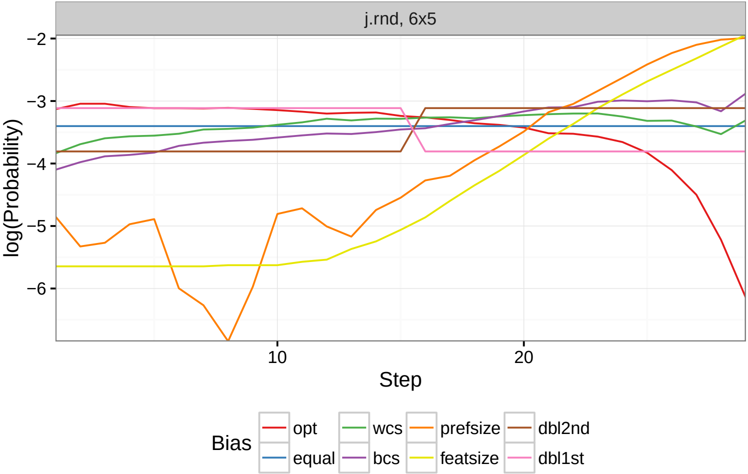
$(6 \times 5, N_{train} = 500)$



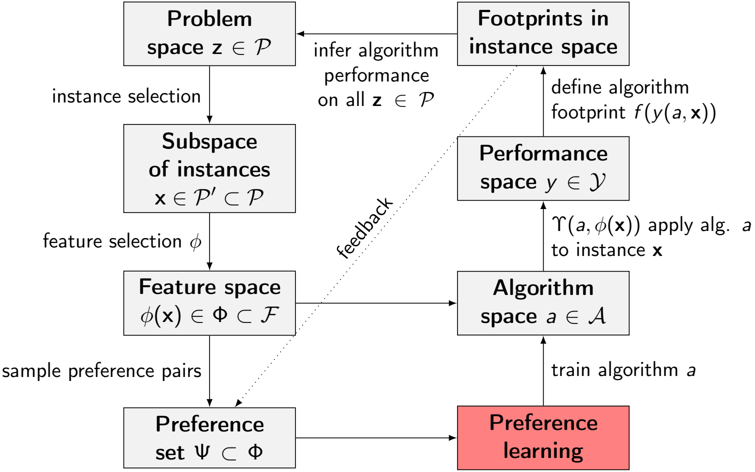
Sampled size of $|\Psi(k)|$
$(6 \times 5, N_{train} = 500)$



Stepwise bias strategies
(6×5, $N_{train} = 500$)



Framework for Algorithm Learning



Preference learning:

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

$$\phi_o \succ \phi_s \iff h(\phi_o) > h(\phi_s)$$

- ⋆ The preference is defined by a linear function:

$$h(\phi) = \langle w_i \cdot \phi \rangle$$

  optimised w.r.t. $\mathbf{w}$ based on training data Ψ

- ⋆ Note: Limitations in approximation function to capture the complex dynamics incorporated in optimal trajectories.

Ordinal Regression

Preference learning:

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

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

- ⋆ The preference is defined by a linear function:

$$h(\phi) = \langle w_i \cdot \phi \rangle$$

optimised w.r.t. $\mathbf{w}$ based on training data Ψ

- ⋆ Note: Limitations in approximation function to capture the complex dynamics incorporated in optimal trajectories.

Ordinal Regression

Preference learning:

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

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

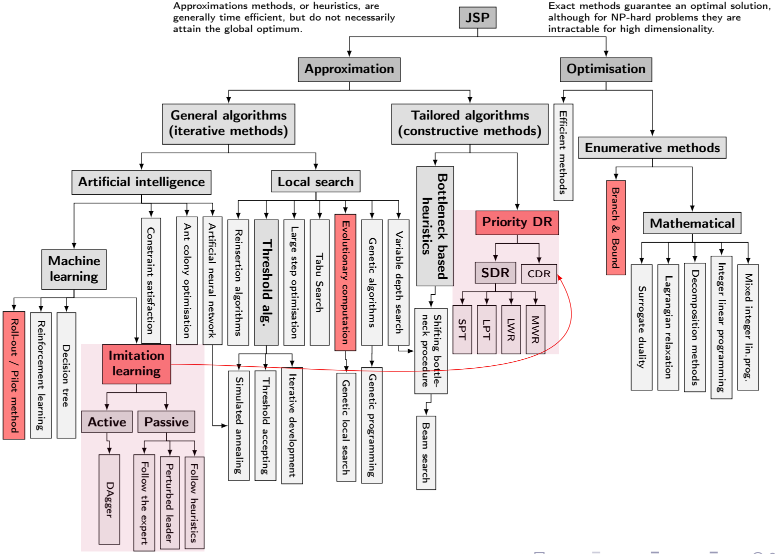
- The preference is defined by a linear function:

$$h(\phi) = \langle w_i \cdot \phi \rangle$$

optimised w.r.t. $\mathbf{w}$ based on training data Ψ

- Note: Limitations in approximation function to capture the complex dynamics incorporated in optimal trajectories.

Various Methods for Solving JSP



Passive imitation learning

Passive imitation learning (single pass):

- Prediction with expert advice, $\pi_\star$
- Follow the perturbed leader (OPTϵ)
- Follow a heuristic (e.g. SDRs).

Passive imitation learning

Passive imitation learning (single pass):

- $\star$ Prediction with expert advice, $\pi_\star$
- $\star$ Follow the perturbed leader (OPTϵ)
- $\star$ Follow a heuristic (e.g. SDRs).

Passive imitation learning

Passive imitation learning (single pass):

- ⋆ Prediction with expert advice, $\pi_\star$
- ⋆ Follow the perturbed leader (OPTϵ)
- ⋆ Follow a heuristic (e.g. SDRs).

Passive imitation learning

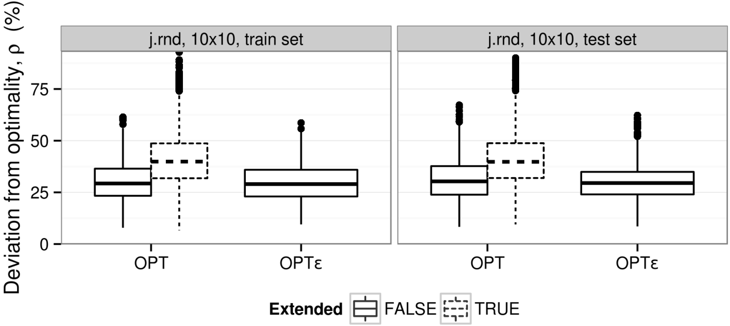
Passive imitation learning (single pass):

- $\star$ Prediction with expert advice, $\pi_\star$
- $\star$ Follow the perturbed leader (OPTϵ)
- $\star$ Follow a heuristic (e.g. SDRs).

Passive imitation learning

Passive imitation learning (single pass):

- ⋆ Prediction with expert advice, $\pi_\star$
- ⋆ Follow the perturbed leader ($\mathrm{OPT}\epsilon$)
- ⋆ Follow a heuristic (e.g. SDRs).



Active imitation learning

Active imitation learning (iterative):

- Dataset Aggregation (DAgger)

$$\pi_i = \beta_i \pi_\star + (1 - \beta_i)\hat{\pi}_{i-1}$$

where $\hat{\pi}_{i-1}$ is the previous learned model, and $\hat{\pi}_i$ learns on aggregated dataset of all previous iterations.

Active imitation learning

Active imitation learning (iterative):

- ⋆ Dataset Aggregation (DAgger)

$$\pi_i = \beta_i \pi_\star + (1 - \beta_i)\hat{\pi}_{i-1}$$

where $\hat{\pi}_{i-1}$ is the previous learned model, and $\hat{\pi}_i$ learns on aggregated dataset of all previous iterations.

Active imitation learning

Active imitation learning (iterative):

- ⋆ Dataset Aggregation (DAgger)

$$\pi_i = \beta_i \pi_\star + (1 - \beta_i)\hat{\pi}_{i-1}$$

where $\hat{\pi}_{i-1}$ is the previous learned model, and $\hat{\pi}_i$ learns on aggregated dataset of all previous iterations.

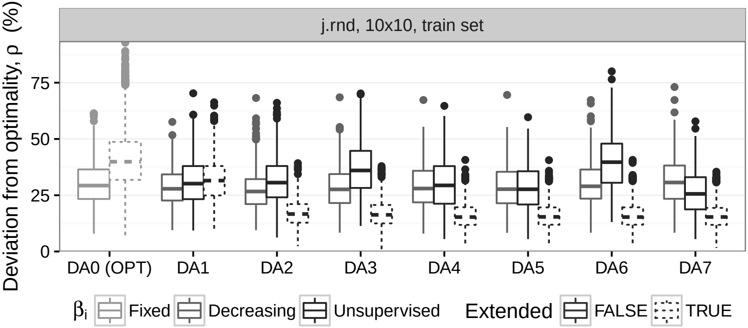
Active imitation learning

Active imitation learning (iterative):

- ⋆ Dataset Aggregation (DAgger)

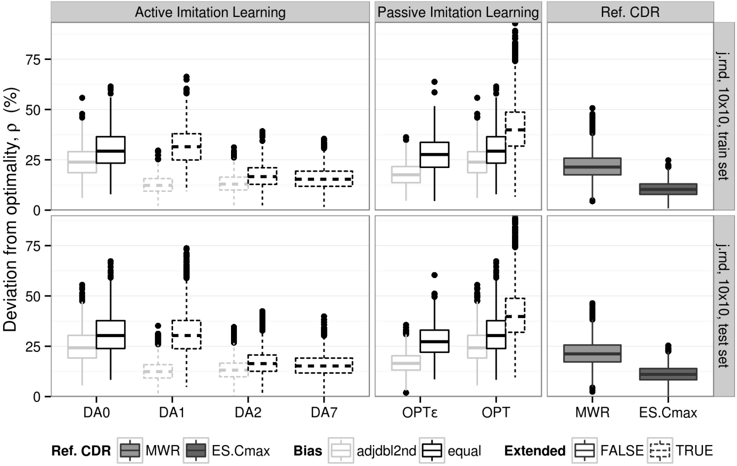
$$\pi_i = \beta_i \pi_\star + (1 - \beta_i)\hat{\pi}_{i-1}$$

where $\hat{\pi}_{i-1}$ is the previous learned model, and $\hat{\pi}_i$ learns on aggregated dataset of all previous iterations.

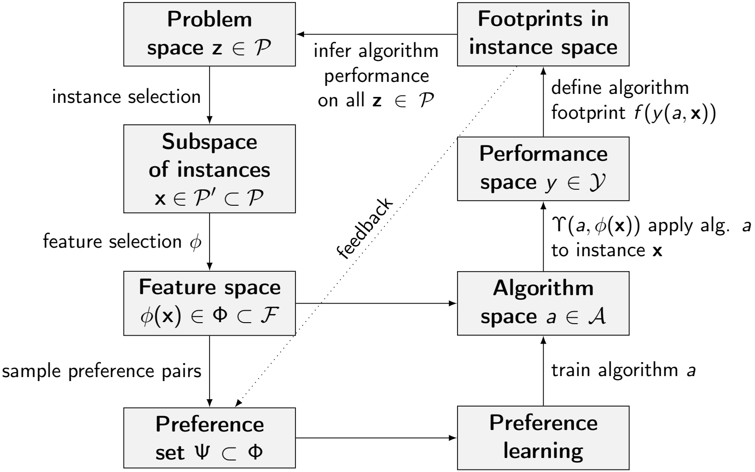


Deviation from optimality, ρ

- ALICE
- Helga
- Introduction
- Problem space
- Subspace of instances
- Feature space
- Algorithm space
- Performance space
- Footprints in instance space
- Preference set
- **Preference learning**
- Conclusions



Framework for Algorithm Learning



Using Analysis & Learning Iterative Consecutive Executions framework I

The thesis introduced a framework for learning (linear) composite priority dispatching rule – using job-shop as a case-study – with the following guidelines:

- For a given problem domain, use a suitable problem generator to train and test on.
- Define features to grasp the essence of visited k-solutions
- Success is highly dependent on the preference pairs introduced to the system:
  - Ψ_p reduces the preference set without loss of performance.
  - Stepwise bias is needed to balance time dependent Ψ_p in order to create time independent models.

It is non intuitive how to go about collecting training data.

Using Analysis & Learning Iterative Consecutive Executions framework I

The thesis introduced a framework for learning (linear) composite priority dispatching rule – using job-shop as a case-study – with the following guidelines:

- ⋆ For a given problem domain, use a suitable problem generator to train and test on.
- ⋆ Define features to grasp the essence of visited k-solutions
- ⋆ Success is highly dependent on the preference pairs introduced to the system:
  - ⋆ Ψ_p reduces the preference set without loss of performance.
  - ⋆ Stepwise bias is needed to balance time dependent Ψ_p in order to create time independent models.

It is non intuitive how to go about collecting training data.

Using Analysis & Learning Iterative Consecutive Executions framework I

The thesis introduced a framework for learning (linear) composite priority dispatching rule – using job-shop as a case-study – with the following guidelines:

- ⋆ For a given problem domain, use a suitable problem generator to train and test on.
- ⋆ Define features to grasp the essence of visited k-solutions
- ⋆ Success is highly dependent on the preference pairs introduced to the system:
  - ⋆ Ψ_p reduces the preference set without loss of performance.
  - ⋆ Stepwise bias is needed to balance time dependent Ψ_p in order to create time independent models.

It is non intuitive how to go about collecting training data.

Using Analysis & Learning Iterative Consecutive Executions framework I

The thesis introduced a framework for learning (linear) composite priority dispatching rule – using job-shop as a case-study – with the following guidelines:

- ⋆ For a given problem domain, use a suitable problem generator to train and test on.
- ⋆ Define features to grasp the essence of visited k-solutions
- ⋆ Success is highly dependent on the preference pairs introduced to the system:
  - ⋆ Ψ_p reduces the preference set without loss of performance.
  - ⋆ Stepwise bias is needed to balance time dependent Ψ_p in order to create time independent models.

It is non intuitive how to go about collecting training data.

Using Analysis & Learning Iterative Consecutive Executions framework I

The thesis introduced a framework for learning (linear) composite priority dispatching rule – using job-shop as a case-study – with the following guidelines:

- ⋆ For a given problem domain, use a suitable problem generator to train and test on.
- ⋆ Define features to grasp the essence of visited k-solutions
- ⋆ Success is highly dependent on the preference pairs introduced to the system:
  - ⋆ Ψ_p reduces the preference set without loss of performance.
  - ⋆ Stepwise bias is needed to balance time dependent Ψ_p in order to create time independent models.

It is non intuitive how to go about collecting training data.

Using Analysis & Learning Iterative Consecutive Executions framework I

The thesis introduced a framework for learning (linear) composite priority dispatching rule – using job-shop as a case-study – with the following guidelines:

- ⋆ For a given problem domain, use a suitable problem generator to train and test on.
- ⋆ Define features to grasp the essence of visited k-solutions
- ⋆ Success is highly dependent on the preference pairs introduced to the system:
  - ⋆ Ψ_p reduces the preference set without loss of performance.
  - ⋆ Stepwise bias is needed to balance time dependent Ψ_p in order to create time independent models.

It is non intuitive how to go about collecting training data.

Using Analysis & Learning Iterative Consecutive Executions framework I

The thesis introduced a framework for learning (linear) composite priority dispatching rule – using job-shop as a case-study – with the following guidelines:

- ⋆ For a given problem domain, use a suitable problem generator to train and test on.
- ⋆ Define features to grasp the essence of visited k-solutions
- ⋆ Success is highly dependent on the preference pairs introduced to the system:
  - ⋆ Ψ_p reduces the preference set without loss of performance.
  - ⋆ Stepwise bias is needed to balance time dependent Ψ_p in order to create time independent models.

It is non intuitive how to go about collecting training data.

Using Analysis & Learning Iterative Consecutive Executions framework II

Continued from prev. slide:

- ⋆ Learning optimal trajectories predominant in literature. Study showed Φ^{OPT} can result in insufficient knowledge.
- ⋆ Following sub-optimal deterministic policies, yet labelling with an optimal solver, improves the guiding policy.
- ⋆ Active update procedure using DAgger ensures sample states the learned model is likely to encounter is integrated to $\Psi_p^{\mathrm{DA}i}$.
- ⋆ Instead of reusing same problem instances, extend the training set with new instances for quicker convergence of DAgger.
- ⋆ In sequential decision making, all future observations are dependent on previous operations.

Using Analysis & Learning Iterative Consecutive Executions framework II

Continued from prev. slide:

- ⋆ Learning optimal trajectories predominant in literature. Study showed Φ^{OPT} can result in insufficient knowledge.
- ⋆ Following sub-optimal deterministic policies, yet labelling with an optimal solver, improves the guiding policy.
- ⋆ Active update procedure using DAgger ensures sample states the learned model is likely to encounter is integrated to $\Psi_p^{\text{DA}i}$.
- ⋆ Instead of reusing same problem instances, extend the training set with new instances for quicker convergence of DAgger.
- ⋆ In sequential decision making, all future observations are dependent on previous operations.

Using Analysis & Learning Iterative Consecutive Executions framework II

Continued from prev. slide:

- ⋆ Learning optimal trajectories predominant in literature. Study showed Φ^{OPT} can result in insufficient knowledge.
- ⋆ Following sub-optimal deterministic policies, yet labelling with an optimal solver, improves the guiding policy.
- ⋆ Active update procedure using DAgger ensures sample states the learned model is likely to encounter is integrated to $\Psi_p^{\mathrm{DA}i}$.
- ⋆ Instead of reusing same problem instances, extend the training set with new instances for quicker convergence of DAgger.
- ⋆ In sequential decision making, all future observations are dependent on previous operations.

Using Analysis & Learning Iterative Consecutive Executions framework II

Continued from prev. slide:

- ⋆ Learning optimal trajectories predominant in literature. Study showed Φ^{OPT} can result in insufficient knowledge.
- ⋆ Following sub-optimal deterministic policies, yet labelling with an optimal solver, improves the guiding policy.
- ⋆ Active update procedure using DAgger ensures sample states the learned model is likely to encounter is integrated to $\Psi_p^{\mathrm{DA}i}$.
- ⋆ Instead of reusing same problem instances, extend the training set with new instances for quicker convergence of DAgger.
- ⋆ In sequential decision making, all future observations are dependent on previous operations.

Using Analysis & Learning Iterative Consecutive Executions framework II

Continued from prev. slide:

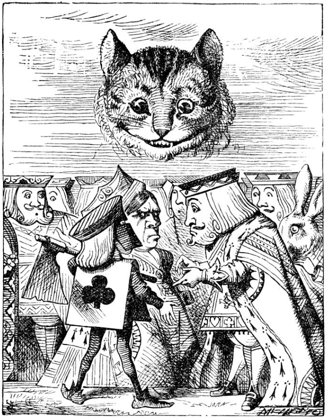
- ⋆ Learning optimal trajectories predominant in literature. Study showed Φ^{OPT} can result in insufficient knowledge.
- ⋆ Following sub-optimal deterministic policies, yet labelling with an optimal solver, improves the guiding policy.
- ⋆ Active update procedure using DAgger ensures sample states the learned model is likely to encounter is integrated to $\Psi_p^{\mathrm{DA}i}$.
- ⋆ Instead of reusing same problem instances, extend the training set with new instances for quicker convergence of DAgger.
- ⋆ In sequential decision making, all future observations are dependent on previous operations.

Acknowledgements

Funding: University of Iceland's Research Fund.

Doctoral committee:

- ⋆ Prof. Tómas Philip Rúnarsson, University of Iceland (advisor).
- ⋆ Prof. Gunnar Stefánsson, University of Iceland.
- ⋆ Prof. Michèle Sebag, Université Paris-Sud.



Thank you for your attention

Questions?

Helga Ingimundardóttir
`hei2@hi.is`

Supplementary material:

- ⋆ Shiny application
- ⋆ Github.