

Ph.D. defense

ALIC

Helga

Introduction

Problem space

Subspace of instances

Feature space

Performance

Footprints in

Preference se

Preference s

......

Conclusion

ALICE

Analysis & Learning Iterative Consecutive Executions

Helga Ingimundardóttir

University of Iceland

June 30, 2016



Motivation:

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



Introduction

ALIC

Helg

Introduction

space Subspace c

instances

Algorithm

Performance space

Footprints in instance space

Preference s

learning

Conclusions

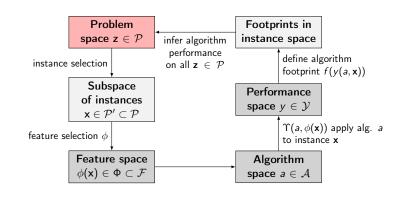
Motivation:

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

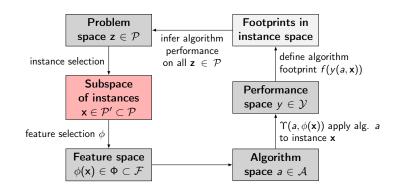
Contribution:

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











ALIC

Helg

Introduction

Problem space

Subspace of instances

instances

Algorithm

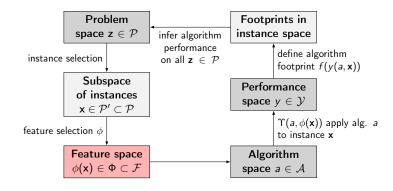
Performance space

Footprints in instance space

Preference se

Preference

C----1---1---





ALIC

Helg

Introduction

Problem space

Subspace of

instances

Algorithm

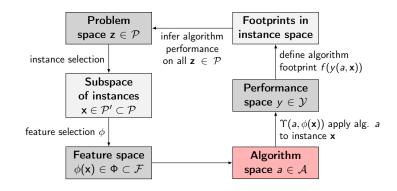
Performance space

Footprints in instance space

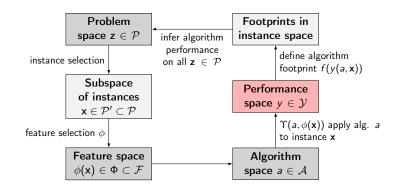
Preference se

Preference

Conclusion









ALIC

Helg

Introduction

Problem space

Subspace of

instances

Algorithm

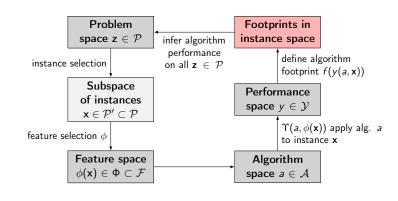
Performance space

Footprints in instance space

Preference se

Preferent learning

onclusio





ALIC

Heig

Introduction

Problem

Subspace of instances

Feature spa

Algorithm

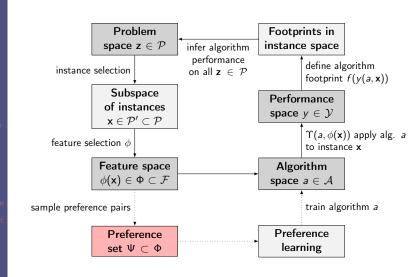
Performance space

Footprints in instance space

Preference s

learning

onclusion





ALIC

Helg

Introduction

Problem

Subspace of

Fastura ens

Algorithm

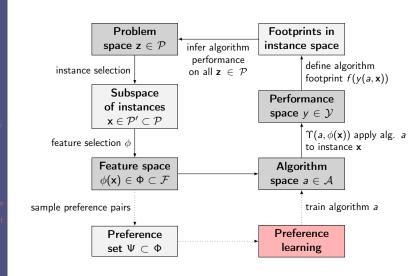
Performance space

Footprints in instance space

Preference s

learning

onclusion





ALIC

Heig

Introduction

Problem

Subspace of

_ .

Algorithm

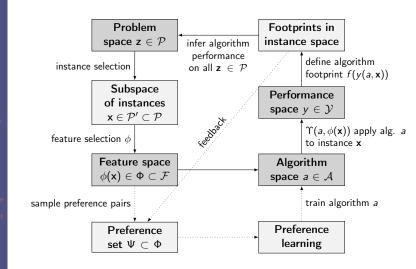
Performance space

Footprints in instance spac

Preterence s

learning

onclusions





ALIC

Helg

Introduction

Problem space

Subspace of

Feature spac

Algorithm

Performance space

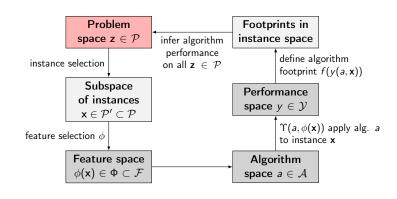
Footprints in instance space

Preference se

i reference s

rearring

onclusio





Job Shop Scheduling (JSP

ALICE

Helg

Introduction

space

instances

Algorithm

Performance space

Footprints in instance space

Preference se

Preferen learning

onclusion

Simple job-shop is where n jobs are scheduled on a set of m machines, subject to constraints:

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

Objective: schedule the jobs so as to minimise the maximum completion time, i.e., makespan, C_{max} .



Dispatching rules (DR) are consecutive executions found by:

- * Starting with an empty schedule and adding on one operation at a time.
- * When a machine is free the DR inspects the available jobs and selects the one 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.



Mad Hatter tea-party (definition)

ALIC

Helg

Probler

Subspace o instances Feature spa

lgorithm

Footprints in Instance spac

Preference earning The attending guests: They all have to:

 J_1) Alice M_1) have wine or pour tea

 J_2) March Hare M_2) spread butter

 J_3) Dormouse M_3) get a haircut

 J_4) Mad Hatter. 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

 \star objective is to minimise C_{max} (when Alice can leave).



Mad Hatter tea-party (action states)

ALIC

Helga

Introduction

Problem space

Subspace of

Feature spa

Algorithm

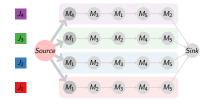
Performance

Footprints in

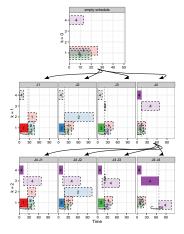
Preference so

Prefere

......









Mad Hatter tea-party (action states)



Helga

Introduction

Problem space

Subspace of

English on

Algorithm

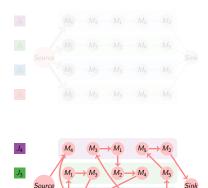
Performance space

Footprints in instance space

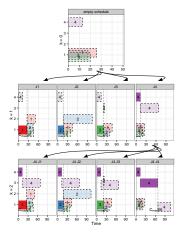
Preference so

Preferen

. . .



 $M_2 \longrightarrow M_3 \longrightarrow M_4 - M_2 \longrightarrow M_2 \longrightarrow M_3 \longrightarrow M_4 - M_4 - M_4 \longrightarrow M_4$





Mad Hatter tea-party (action states)



Helga

Introduction

Problem space

Subspace o

Feature space

reacure space

Algorithm

Performanc space

Footprints in instance space

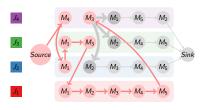
Preference s

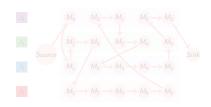
D (

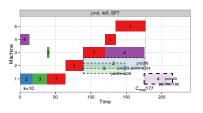
......

onclusion











Mad Hatter tea-party (SDRs)

ALIC

Helga

Introduction

Problem space

Subspace of instances

Feature spa

Algorithm

Performanc space

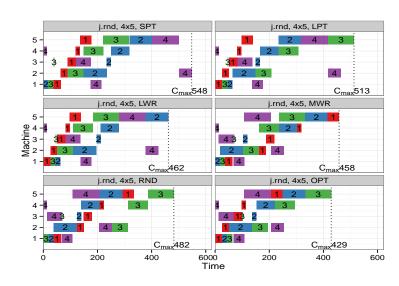
Footprints in instance space

Preference s

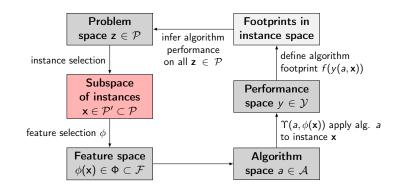
Preference

_ . .

Conclusion









Problem instance generators

ALIC

Helga

ntroduction Problem

Subspace of instances

Feature spa

Algorithm

Performance space

Footprints in instance space

Preference s

Prefere learning

onclusion

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



ALIC

Helg

Introduction Problem

Subspace of

Feature space

Algorithm

Performance space

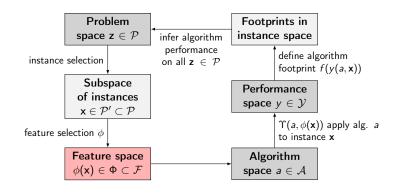
Footprints in instance space

Preference s

Preference

_ . .

onclusio





doį	$egin{array}{c} \phi_1 \ \phi_2 \ \phi_3 \ \phi_4 \ \phi_5 \ \phi_6 \ \phi_7 \ \phi_8 \end{array}$	job processing time job start-time job end-time job arrival time time job had to wait total processing time for job total work remaining for job number of assigned operations for job			
machine	$egin{array}{c} \phi_9 \ \phi_{10} \ \phi_{11} \ \phi_{12} \ \phi_{13} \ \phi_{14} \ \phi_{15} \ \phi_{16} \ \end{array}$	when machine is next free total processing time for machine total work remaining for machine number of assigned operations for machine change in idle time by assignment total idle time for machine total idle time for all machines current makespan			
final makespan	ϕ_{17} ϕ_{18} ϕ_{19} ϕ_{20} ϕ_{RND} ϕ_{21} ϕ_{22} ϕ_{23} ϕ_{24}	final makespan using SPT final makespan using LPT final makespan using LWR final makespan using MWR final makespans using 100 random rollouts mean for $\phi_{\rm RND}$ standard deviation for $\phi_{\rm RND}$ minimum value for $\phi_{\rm RND}$ maximum value for $\phi_{\rm RND}$			



Trajectory strategies for 4

ALIC

Helg

Introduction

Problem space

Subspace o instances

Footure one

Algorithm

Performance

Footprints in instance space

Preference se

Prefere

Conclusion

- $\star (\Phi^{OPT})$ expert π_{\star} .
- \star ($\Phi^{\text{ES},\rho}$) the policy obtained by optimising with CMA-ES.
- \star (Φ^{SPT}) shortest processing time (SPT).
- * (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 (Φ^{ALL}) union of all of the above.



Trajectory strategies for 4

ALIC

Helg

Introduction

Problem space

Subspace of instances

Feature space

Performance

Footprints in instance space

Preference se

Prefere

Canalusias

- \star (Φ^{OPT}) expert π_{\star} .
- * $(\Phi^{ES.\rho})$ the policy obtained by optimising with CMA-ES.
- \star (Φ^{SPT}) shortest processing time (SPT).
- * (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 (Φ^{ALL}) union of all of the above.



Trajectory strategies for Φ

ALIC

Helg

Introduction

space

Subspace of Instances

eature space

Macrithm

Performance

Footprints in instance space

Preference se

Preferent learning

onclusion

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



- \star (Φ^{OPT}) expert π_{\star} .
- \star ($\Phi^{ES.\rho}$) the policy obtained by optimising with CMA-ES.
- \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 (Φ^{ALL}) union of all of the above.



- \star (Φ^{OPT}) expert π_{\star} .
- \star ($\Phi^{ES.\rho}$) the policy obtained by optimising with CMA-ES.
- \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 (Φ^{ALL}) union of all of the above.



Trajectory strategies for Φ

ALIC

Helg

Introductior Problem

Subspace o

nstances

-eature space

Performance space

instance spac

Preference learning

Conclusions

- $\star (\Phi^{OPT})$ expert π_{\star} .
- \star ($\Phi^{\text{ES}.\rho}$) the policy obtained by optimising with CMA-ES.
- \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 (Φ^{ALL}) union of all of the above.



- \star (Φ^{OPT}) expert π_{\star} .
- \star ($\Phi^{ES.\rho}$) the policy obtained by optimising with CMA-ES.
- \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 (Φ^{ALL}) union of all of the above.



Trajectory strategies for Φ

ALICE

Helg

ntroductio Problem space

Subspace of instances

eature space

Performance space

nstance space Preference se Preference

learning Conclusions

- \star (Φ^{OPT}) expert π_{\star} .
- $\star~(\Phi^{\text{ES}.\rho})$ the policy obtained by optimising with CMA-ES.
- \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 (Φ ^{ALL}) union of all of the above.



Sampled size of $|\Phi|$ (6 × 5, $N_{train} = 500$)

ALIC

Helg

Introduction

Problem space

Subspace of instances

Feature space

Algorithm

Performance

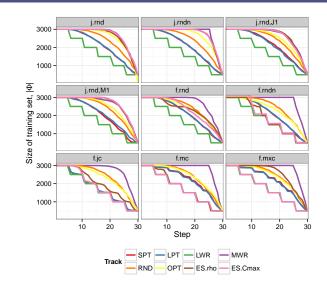
Footprints in

instance space

.

learning

onclusion





ALIC

Helg

Introduction Problem

Subspace of

instances _

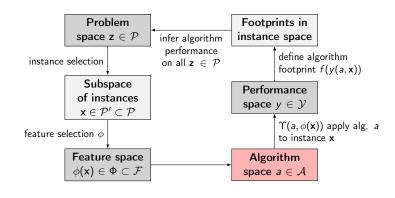
Algorithm

Performance

Footprints in instance space

Preference s

Preference





Various Methods for Solving JSP

ALIC

Helg

Introduction

Problem

Subspace of instances

Algorithm

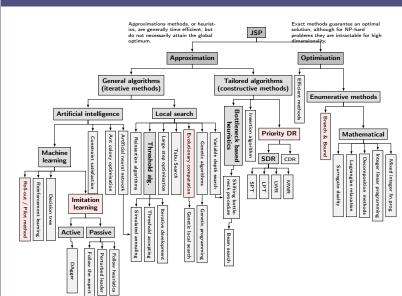
Performanc

space

Preference se

Preference learning

Conclusion





ALIC

Helg

Introduction Problem

Subspace of

Algorithm space

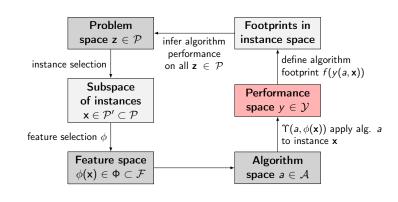
Performance space

Footprints in instance space

Preference s

Preference learning

......





Performance measure

ALIC

Helg

Introduction Problem

Subspace of

Feature spa

Algorithm space

Performance space

instance space

Preference s

Prefere learning

onclusion

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

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

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



Framework for Algorithm Learning

ALIC

Helg

Introduction

Problem space

Subspace of instances

Feature spa

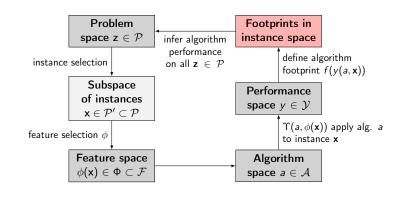
Algorithm

Performance space

Footprints in instance space

Preference s

Preference





Deviation from optimality, /

ALIC

Helga

Introductio

space

instances

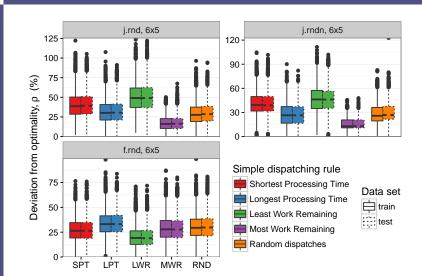
Algorithm

Performance space

Footprints in instance space

Preference s

Preference learning





Deviation from optimality, /

ALIC

Helga

Introduction

Problem space

Subspace o instances

instances

Algorithm

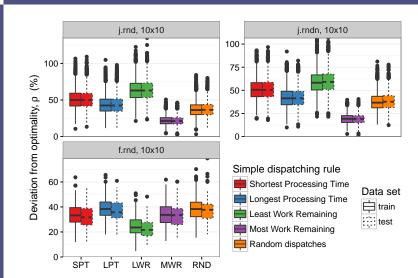
Performance

Footprints in instance space

Preference so

Dueference

learning





Making optimal decisions, ξ

ALIC

Helg

Introduction

Problem

Subspace of

reature spa

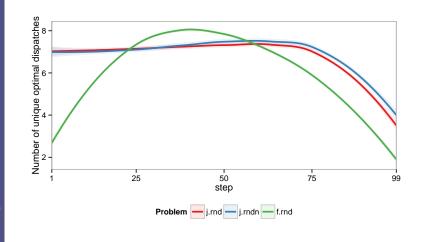
Algorithm

Performance space

Footprints in instance space

Preference s

Prefere





Making optimal decisions, ξ

ALIC

Helg

Introduction

Problem

Subspace of

.

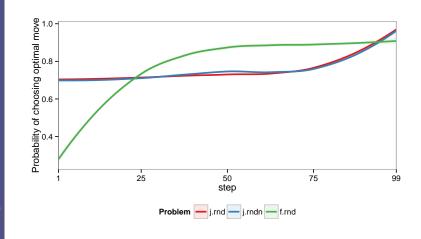
Algorithm

Performance space

Footprints in instance space

Preference s

Prefere





Making optimal decisions, ξ

ALIC

Helg

Introduction

Problem space

Subspace of instances

Feature spa

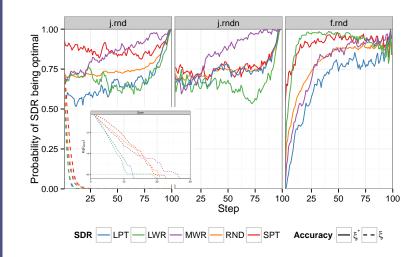
Algorithm space

Performance space

Footprints in instance space

Preference s

Prefere learning





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

ALICI

Helg

Introduction

Problem

Subspace o

Enatura cna

Algorithm

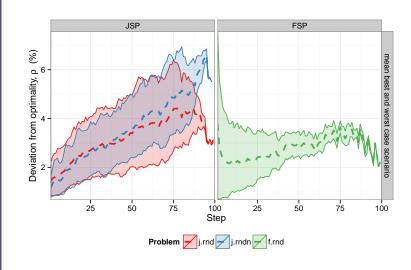
space

Performance space

Footprints in instance space

Preference s

Prefere learning





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

ALICI

Helg

Introduction

Problem

Subspace of

Feature spa

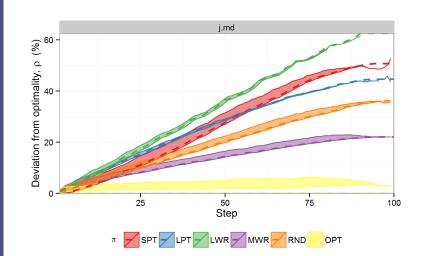
Algorithm

Performance

Footprints in

Preference s

Preference





Blended dispatching rule

ALIC

Helga

ntroduction

Problem space

Subspace o instances

Feature spa

Algorithm space

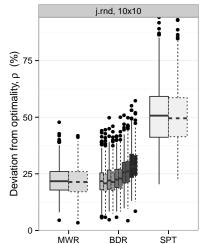
Performance space

Footprints in instance space

Preference so

Prefere learning

onclusion



Dispatching rule

Shortest Processing Time

Most Work Remaining

SPT (first 10 %), MWR (last 90 %)

SPT (first 15 %), MWR (last 85 %)

SPT (first 20 %), MWR (last 80 %)

SPT (first 30 %), MWR (last 70 %)

SPT (first 40 %), MWR (last 60 %)

Data set

train 🚃 test



Framework for Algorithm Learning

ALIC

Helg

Introductic Problem

Subspace of instances

instances

Algorithm

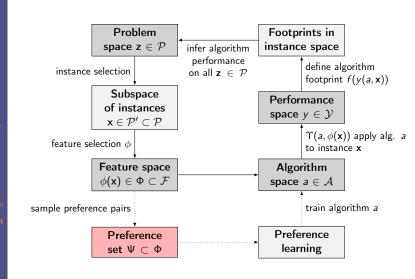
space

space

instance spa

Preference se

Preferer learning





ALICE framework for creating dispatching rules:

- * Linear classification to identify good dispatches, from worse ones.

- \star Sample Φ to create training set Ψ with rank pairs:
 - \star optimal decision, $(z^{\circ}, y_{\circ}) = (\phi^{\circ} \phi^{\circ}, +1)$
 - * suboptimal decision, $(\mathbf{z}^s, y_s) = (\phi^s \phi^o, -1)$

 \star Sample Ψ using stepwise bias for time independent policy.



Generating training data

ALIC

Helg

Introduction Problem

Subspace of instances

-eature spa Algorithm

Performanc

Footprints in instance space

Preference se

learning

Conclusions

ALICE framework for creating dispatching rules:

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

by exploring various trajectories within the feature-space (where $\phi^o, \phi^s \in \mathcal{F}$).

- \star Sample Φ to create training set Ψ with rank pairs:
 - \star optimal decision, $(\mathbf{z}^o, y_o) = (\phi^o \phi^s, +1)$
 - \star suboptimal decision, $(\mathbf{z}^s, y_s) = (\phi^s \phi^o, -1)$

using different ranking schemes (where $z^o,z^s\in \Psi)$

 \star Sample Ψ using stepwise bias for time independent policy.



Generating training data

ALICE

Helg

Problem space

nstances

lgorithm

Performance space

instance spac

Preference learning ALICE framework for creating dispatching rules:

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

by exploring various trajectories within the feature-space (where $\phi^o, \phi^s \in \mathcal{F}$).

- \star Sample Φ to create training set Ψ with rank pairs:
 - \star 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 $\mathbf{z}^o,\mathbf{z}^s\in\Psi)$
- \star Sample Ψ using stepwise bias for time independent policy.



Generating training data

ALICE

Helg

Problem space

nstances

lgorithm pace

space Footprints in Instance space

Preference se

Preference earning
Conclusions

ALICE framework for creating dispatching rules:

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

by exploring various trajectories within the feature-space (where $\phi^o, \phi^s \in \mathcal{F}$).

- \star Sample Φ to create training set Ψ with rank pairs:
 - \star 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.



Ranking schemes for V

ALIC

Helg

Introduction

Problem space

Subspace of instances

Feature sp

Algorithm space

Performance space

Footprints in instance space

Preference se

Preferenc learning

onclusion

Sampling rankings of available jobs where where

$$r_1 > r_2 > \cdots > r_{n'} \ (n' \le n)$$
 with respect to

- Ψ_b all opt rankings r_1 vs. all possible subopt rankings r_i , $i \in \{2, ..., n'\}$
- Ψ_f full subsequent rankings, i.e., all combinations of r_i and r_{i+1} for all $i \in \{1, ..., n'\}$.
- Ψ_p partial subsequent rankings, similar to Ψ_f except if there are more than one operation with the same rank, only one is needed to be compared to subsequent rank $(\Psi_p \subset \Psi_f)$.
- Ψ_a union of all of the above.



Ranking schemes for V

ALIC

Helg

Introduction Problem

Subspace of instances

Algorithm space

space

Preference se

Preference

learning

Sampling rankings of available jobs where where

 $r_1 > r_2 > \cdots > r_{n'} \ (n' \le n)$ with respect to:

 Ψ_b all opt rankings r_1 vs. all possible subopt rankings r_i , $i \in \{2, ..., n'\}$

 Ψ_f full subsequent rankings, i.e., all combinations of r_i and r_{i+1} for all $i \in \{1, ..., n'\}$.

 Ψ_p partial subsequent rankings, similar to Ψ_f except if there are more than one operation with the same rank, only one is needed to be compared to subsequent rank $(\Psi_p \subset \Psi_f)$.

 Ψ_a union of all of the above.



Sampling rankings of available jobs where where

 $r_1 > r_2 > \cdots > r_{n'}$ $(n' \le n)$ with respect to:

 Ψ_{h} all opt rankings r_1 vs. all possible subopt rankings r_i , $i \in \{2, ..., n'\}$

 Ψ_f full subsequent rankings, i.e., all combinations of r_i and r_{i+1} for all $i \in \{1, ..., n'\}$.



Ranking schemes for V

ALIC

Helg

Introduction Problem

instances
Feature space
Algorithm
space

space Footprints in instance spa

Preference se

Preterence learning Sampling rankings of available jobs where where

 $r_1 > r_2 > \cdots > r_{n'} \ (n' \le n)$ with respect to:

 Ψ_b all opt rankings r_1 vs. all possible subopt rankings r_i , $i \in \{2, ..., n'\}$

 Ψ_f full subsequent rankings, i.e., all combinations of r_i and r_{i+1} for all $i \in \{1, ..., n'\}$.

 Ψ_p partial subsequent rankings, similar to Ψ_f except if there are more than one operation with the same rank, only one is needed to be compared to subsequent rank $(\Psi_p \subset \Psi_f)$.

 Ψ_a union of all of the above.



Ranking schemes for V

ALIC

Helg

ntroductio Problem space

Subspace of instances Feature spac

Performance space

instance space

Preference se

Preterence learning Conclusions Sampling rankings of available jobs where where

 $r_1 > r_2 > \cdots > r_{n'} \ (n' \le n)$ with respect to:

 Ψ_b all opt rankings r_1 vs. all possible subopt rankings r_i , $i \in \{2, ..., n'\}$

 Ψ_f full subsequent rankings, i.e., all combinations of r_i and r_{i+1} for all $i \in \{1, ..., n'\}$.

 Ψ_p partial subsequent rankings, similar to Ψ_f except if there are more than one operation with the same rank, only one is needed to be compared to subsequent rank $(\Psi_p \subset \Psi_f)$.

 Ψ_a union of all of the above.



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

ALIC

Helg

ntroduction

Problem space

Subspace o

Footune on

Algorithm

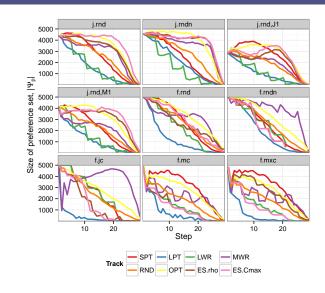
Algorithm space

Space space

instance spa

Preterence s

learning





ALIC

нец

Introduction

Problem space

Subspace of instances

. Algorithm

Performance

Footprints in instance space

Preference se

Preference learning

`analusias

- * (equal) equal probability.
- \star (opt) inverse optimality for random dispatches $1-\xi_{\mathsf{RND}}^{\star}$.
- \star (bcs) best case scenario for mean $\rho \zeta_{\min}^{\star}$.
- \star (wcs) worst case scenario for mean $\rho \zeta_{\max}^{\star}$.
- \star (featsize) inversely proportional to $|\Phi^{OPT}|$
- \star (prefsize) inversely proportional to $|\Psi_{n}^{OPT}|$
- * (dbl1st) twice as much weight on the first half of the dispatches.
- * (dbl2nd) twice as much weight on the second half of the dispatches.



ALIC

Helg

Introduction

Subspace o

instances

Algorithm

Performance

Footprints in instance space

Preference se

Preference learning

onclusion

- * (equal) equal probability.
- \star (opt) inverse optimality for random dispatches $1-\xi_{\mathsf{RND}}^{\star}$.
- \star (bcs) best case scenario for mean $\rho \zeta_{\min}^{\star}$.
- \star (wcs) worst case scenario for mean ρ $\zeta_{\rm max}^{\star}$.
- \star (featsize) inversely proportional to $|\Phi^{OPT}|$
- \star (prefsize) inversely proportional to $|\Psi_p^{OPT}|$
- * (dbl1st) twice as much weight on the first half of the dispatches.
- (dbl2nd) twice as much weight on the second half of the dispatches.



ALIC

Helg

Introduction

Subspace o

instances

Feature spa

Performance

Footprints in instance space

Preference se

Preterence learning

onclusions

- * (equal) equal probability.
- \star (opt) inverse optimality for random dispatches $1-\xi_{\mathsf{RND}}^{\star}$.
- \star (bcs) best case scenario for mean $\rho \zeta_{\min}^{\star}$.
- \star (wcs) worst case scenario for mean $\rho \zeta_{\max}^{\star}$.
- \star (featsize) inversely proportional to $|\Phi^{OPT}|$
- \star (prefsize) inversely proportional to $|\Psi_p^{OPT}|$
- * (dbl1st) twice as much weight on the first half of the dispatches.
- * (dbl2nd) twice as much weight on the second half of the dispatches.



Stepwise bias for sampling \(\Psi \)

ALIC

Helg

Introduction Problem

Subspace of

Instances

Algorithm

Performance space

Footprints in instance space

Preference se

Preferer learning

onclusion

- * (equal) equal probability.
- \star (opt) inverse optimality for random dispatches $1-\xi_{\mathsf{RND}}^{\star}$.
- \star (bcs) best case scenario for mean $\rho \zeta_{\min}^{\star}$.
- * (wcs) worst case scenario for mean $\rho \zeta_{\text{max}}^{\star}$.
- \star (featsize) inversely proportional to $|\Phi^{OPT}|$
- \star (prefsize) inversely proportional to $|\Psi_p^{OPT}|$
- * (dbl1st) twice as much weight on the first half of the dispatches.
- * (dbl2nd) twice as much weight on the second half of the dispatches.



Stepwise bias for sampling \(\Psi \)

ALIC

Helg

Introduction Problem

Subspace of instances

Feature spa

Algorithm space

Performance space

Footprints in instance space

Preference se

Preferer learning

^onclusior

- * (equal) equal probability.
- \star (opt) inverse optimality for random dispatches $1-\xi_{\mathsf{RND}}^{\star}$.
- \star (bcs) best case scenario for mean $\rho \zeta_{\min}^{\star}$.
- \star (wcs) worst case scenario for mean $\rho \zeta_{\text{max}}^{\star}$.
- * (featsize) inversely proportional to $|\Phi^{OPT}|$
- \star (prefsize) inversely proportional to $|\Psi_p^{\mathsf{OPT}}|$
- * (dbl1st) twice as much weight on the first half of the dispatches.
- * (dbl2nd) twice as much weight on the second half of the dispatches.



- * (equal) equal probability.
- \star (opt) inverse optimality for random dispatches 1 ξ_{RND}^{\star} .
- \star (bcs) best case scenario for mean $\rho \zeta_{\min}^{\star}$.
- \star (wcs) worst case scenario for mean $\rho \zeta_{\text{max}}^{\star}$.
- \star (featsize) inversely proportional to $|\Phi^{OPT}|$
- \star (prefsize) inversely proportional to $|\Psi_{n}^{OPT}|$
- * (dbl1st) twice as much weight on the first half of the



Stepwise bias for sampling \(\Psi \)

ALIC

Helg

Introduction
Problem

Subspace of instances

reature spa Algorithm

Performance space

Footprints in instance spac

Preference se

learning Conclusion

- * (equal) equal probability.
- \star (opt) inverse optimality for random dispatches $1-\xi_{\mathsf{RND}}^{\star}$.
- \star (bcs) best case scenario for mean $\rho \zeta_{\min}^{\star}$.
- \star (wcs) worst case scenario for mean $\rho \zeta_{\max}^{\star}$.
- \star (featsize) inversely proportional to $|\Phi^{\mathsf{OPT}}|$
- \star (prefsize) inversely proportional to $|\Psi_p^{OPT}|$
- * (dbl1st) twice as much weight on the first half of the dispatches.
- * (dbl2nd) twice as much weight on the second half of the dispatches.



- * (equal) equal probability.
- \star (opt) inverse optimality for random dispatches 1 ξ_{RND}^{\star} .
- \star (bcs) best case scenario for mean $\rho \zeta_{\min}^{\star}$.
- \star (wcs) worst case scenario for mean $\rho \zeta_{\text{max}}^{\star}$.
- \star (featsize) inversely proportional to $|\Phi^{OPT}|$
- \star (prefsize) inversely proportional to $|\Psi_n^{OPT}|$
- * (dbl1st) twice as much weight on the first half of the dispatches.



ALIC

Helg

ntroduction Problem space

Subspace o instances

eature spa Algorithm

Performance space

nstance spac

Preference se

Preference learning Conclusion

- ★ (equal) equal probability.
- \star (opt) inverse optimality for random dispatches $1-\xi_{\mathsf{RND}}^{\star}$.
- \star (bcs) best case scenario for mean $\rho \zeta_{\min}^{\star}$.
- \star (wcs) worst case scenario for mean $\rho \zeta_{\max}^{\star}$.
- \star (featsize) inversely proportional to $|\Phi^{\mathsf{OPT}}|$
- \star (prefsize) inversely proportional to $|\Psi_p^{\mathsf{OPT}}|$
- ★ (dbl1st) twice as much weight on the first half of the dispatches.
- * (dbl2nd) twice as much weight on the second half of the dispatches.



Stepwise bias strategies

ALIC

Helg

Introduction

Problem space

Subspace o instances

Feature spa

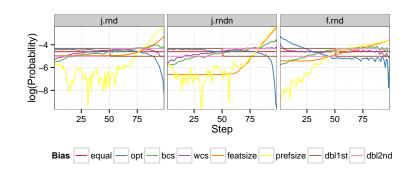
Algorithm

Performanc space

Footprints in instance space

Preference s

Preference learning





Framework for Algorithm Learning

ALIC

Helg

ntroductio Problem

Subspace o instances

Feature spa

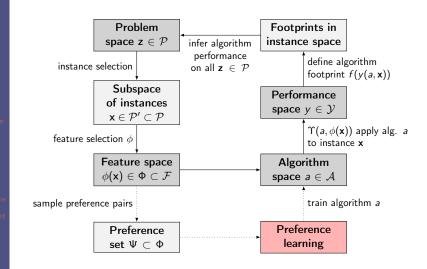
Algorithm

Performance space

Footprints in instance space

Preference s

icuiting





Ordinal Regression

ALIC

Helg

Introduction

space

instances

Algorithm

space

space

Footprints in instance space

Preference s

Preferer learning

onclusion

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:

$$h(\phi) = \sum_{i=1}^{d} w_i \phi$$

optimised w.r.t. w based on training data Ψ

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



Ordinal Regression

ALIC

Helg

Introduction

Subspace of

instances

Algorithm

space

Performance space

Footprints in instance space

Preference s

Preference learning

onclusions

Preference learning:

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

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

* The preference is defined by a linear function:

$$h(\phi) = \sum_{i=1}^{d} \mathbf{w}_{i} \phi$$

optimised w.r.t. w based on training data Ψ

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

Preference learning:

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

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

* The preference is defined by a linear function:

$$h(\phi) = \sum_{i=1}^d w_i \phi$$

optimised w.r.t. w based on training data Ψ

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



Learning methods

ALIC

Helg

Introduction

space

Subspace o[.] instances

Feature spa

Algorithm

Performance space

Footprints in instance space

Preference se

Prefe learni

Conclusior

Passive imitation learning (single pass):

- * Prediction with expert advice, π_* Gurobi
- \star Follow the perturbed leader (OPT ϵ).

Active imitation learning (iterative):

* Dataset Aggregation (DAgger)

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

where $\hat{\pi}_{i-1}$ is the previous learned model, and $\hat{\pi}_i$ learns on

$$\Phi^{\mathsf{DA}i} = \bigcup_{i'=0}^{i} \Phi^{\mathsf{IL}i}$$

aggregated dataset of all previous iterations



Passive imitation learning (single pass):

- * Prediction with expert advice, π_* Gurobi
- * Follow the perturbed leader (OPT_{ϵ}).

* Dataset Aggregation (DAgger)

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

$$\Phi^{\mathsf{DA}i} = \bigcup_{i'=0}^{i} \Phi^{\mathsf{IL}i'}$$



Learning methods

ALIC

Helg

Introductior Problem

Subspace of instances

Feature spa Algorithm

Performance space

Footprints in instance spac

Preference learning

Conclusion

Passive imitation learning (single pass):

- * Prediction with expert advice, π_* Gurobi
- * Follow the perturbed leader (OPT ϵ).

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

$$\Phi^{\mathsf{DA}i} = \bigcup_{i'=0}^{i} \Phi^{\mathsf{IL}i'}$$

aggregated dataset of all previous iterations



Learning methods

ALIC

Helg

Introduction Problem

Subspace of instances

Feature spa Algorithm

Performance space

Footprints in instance space

Preference learning

Conclusion

Passive imitation learning (single pass):

- * Prediction with expert advice, π_{\star} Gurobi
- * Follow the perturbed leader (OPT ϵ).

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

$$\Phi^{\mathsf{DA}i} = \bigcup_{i'=0}^{i} \Phi^{\mathsf{IL}i'}$$

aggregated dataset of all previous iterations.



ALIC

Helg

Introduction

Problem space

Subspace o[.] instances

Feature ens

Algorithm

Performance space

Footprints in instance space

Preference se

Prefere

Conclusio

```
Require: T \ge 1
```

```
1: procedure DAgger(\pi_{\star}, \Phi^{\mathsf{IL0}}, T)
```

2:
$$\hat{\pi}_0 \leftarrow \text{Train}(\Phi^{\text{IL}0})$$
 \triangleright initial model, iff $\Phi^{\text{IL}0} = \Phi^{\text{OPT}}$

3: **for** $i \leftarrow 1$ **to** T **do** \triangleright at each imitation learning iteration

Let
$$\pi_i = \beta_i \pi_* + (1 - \beta_i) \tilde{\pi}_{i-1}$$

5: Sample a
$$K$$
-solution using $\pi_i
ightharpoonup IL(i, \hat{\pi}_{i-1}, \pi_*)$

6:
$$\Phi^{\mathsf{IL}i} = \{(s, \pi_{\star}(s))\} \triangleright \text{ visited by } \pi_i \text{ and actions by } \pi$$

7:
$$\Phi^{DAi} \leftarrow \Phi^{DAi-1} \cup \Phi^{ILi} \qquad \triangleright \text{ aggregate datasets}$$

8:
$$\hat{\pi}_{i+1} \leftarrow \mathsf{Train}(\Phi^{\mathsf{DA}i})$$
 \triangleright preference mode

0: **return** best
$$\hat{\pi}_i$$
 on validation \triangleright best preference model



ALIC

Helg

Introduction

Problem space

Subspace of instances

Feature spa

Algorithm

Performance space

Footprints in instance space

Preference se

Dueference

Conclusio

Require: $T \ge 1$

```
1: procedure DAgger(\pi_{\star}, \Phi^{\mathsf{IL0}}, T)
```

2:
$$\hat{\pi}_0 \leftarrow \text{Train}(\Phi^{\text{IL}0})$$
 \Rightarrow initial model, iff $\Phi^{\text{IL}0} = \Phi^{\text{OPT}}$

3: **for** $i \leftarrow 1$ **to** T **do** \triangleright at each imitation learning iteration

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

5: Sample a
$$K$$
-solution using $\pi_i
ightharpoonup IL(i, \hat{\pi}_{i-1}, \pi_*)$

6:
$$\Phi^{\mathsf{IL}i} = \{(s, \pi_{\star}(s))\} \triangleright \text{ visited by } \pi_i \text{ and actions by } \pi_i$$

7:
$$\Phi^{DAi} \leftarrow \Phi^{DAi-1} \cup \Phi^{ILi} \qquad \triangleright \text{ aggregate datasets}$$

8:
$$\hat{\pi}_{i+1} \leftarrow \mathsf{Train}(\Phi^{\mathsf{DA}i})$$
 \triangleright preference mode

0: **return** best
$$\hat{\pi}_i$$
 on validation \triangleright best preference model



ALIC

Helg

Introduction

Problem space

Subspace of instances

Feature spa

Algorithm

Performance space

Footprints in instance space

Preference se

Dueference

Conclusio

```
Require: T \ge 1
```

```
1: procedure DAgger(\pi_{\star}, \Phi^{\mathsf{IL0}}, T)
```

2:
$$\hat{\pi}_0 \leftarrow \text{Train}(\Phi^{\text{IL}0})$$
 \Rightarrow initial model, iff $\Phi^{\text{IL}0} = \Phi^{\text{OPT}}$

3: **for** $i \leftarrow 1$ **to** T **do** \triangleright at each imitation learning iteration

4: Let
$$\pi_i = \beta_i \pi_* + (1 - \beta_i) \hat{\pi}_{i-1}$$

5: Sample a
$$K$$
-solution using $\pi_i
ightharpoonup IL(i, \hat{\pi}_{i-1}, \pi_*)$

6:
$$\Phi^{\mathsf{IL}i} = \{(s, \pi_{\star}(s))\} \triangleright \text{ visited by } \pi_i \text{ and actions by } \pi$$

7:
$$\Phi^{\mathsf{DA}i} \leftarrow \Phi^{\mathsf{DA}i-1} \cup \Phi^{\mathsf{IL}i}$$
 \triangleright aggregate datasets

8:
$$\hat{\pi}_{i+1} \leftarrow \mathsf{Train}(\Phi^{\mathsf{DA}i})$$
 \triangleright preference model

.0: **return** best
$$\hat{\pi}_i$$
 on validation \triangleright best preference model



ALIC

Helg

Introduction

Problem space

Subspace of instances

Feature spa

Algorithm

Performance space

Footprints in instance space

Preference s

Prefer learnin

Conclusion

```
Require: T \ge 1
```

```
1: procedure DAgger(\pi_{\star}, \Phi^{\mathsf{IL0}}, T)
```

2:
$$\hat{\pi}_0 \leftarrow \text{Train}(\Phi^{\text{IL}0})$$
 \triangleright initial model, iff $\Phi^{\text{IL}0} = \Phi^{\text{OPT}}$

3: **for** $i \leftarrow 1$ **to** T **do** \triangleright at each imitation learning iteration

4: Let
$$\pi_i = \beta_i \pi_* + (1 - \beta_i) \hat{\pi}_{i-1}$$

5: Sample a
$$K$$
-solution using $\pi_i
ightharpoonup IL $(i, \hat{\pi}_{i-1}, \pi_{\star})$$

6:
$$\Phi^{\text{IL}i} = \{(s, \pi_*(s))\} \triangleright \text{ visited by } \pi_i \text{ and actions by } \pi_*$$

7:
$$\Phi^{DAI} \leftarrow \Phi^{DAI-1} \cup \Phi^{ILI}$$
 \triangleright aggregate datasets

8:
$$\hat{\pi}_{i+1} \leftarrow \text{Train}(\Phi^{DAi})$$
 \triangleright preference model

10: **return** best
$$\hat{\pi}_i$$
 on validation ho best preference mode

11: end procedure



ALIC

Helg

Introduction

Subspace of

- .

Algorithm

Space Performance

Footprints in instance space

Preference se

Prefere

Conclusion

```
Require: T \ge 1

1: procedure DAgger(\pi_*, \Phi^{\mathsf{IL0}}, T)

2: \hat{\pi}_0 \leftarrow \mathsf{Train}(\Phi^{\mathsf{IL0}}) \triangleright initial model, iff \Phi^{\mathsf{IL0}} = \Phi^{\mathsf{OPT}}

3: for i \leftarrow 1 to T do \triangleright at each imitation learning iteration
```

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

5: Sample a K-solution using $\pi_i
ightharpoonup IL(i, \hat{\pi}_{i-1}, \pi_{\star})$

6: $\Phi^{\text{IL}i} = \{(s, \pi_{\star}(s))\} \triangleright \text{ visited by } \pi_i \text{ and actions by } \pi_{\star}$

7: $\Phi^{DAi} \leftarrow \Phi^{DAi-1} \cup \Phi^{DAi}$ > aggregate dataset

8: $\hat{\pi}_{i+1} \leftarrow \text{Train}(\Phi^{\text{DA}i})$ \triangleright preference mode

9: end for

10: **return** best $\hat{\pi}_i$ on validation \triangleright best preference model

11: end procedure



```
Require: T > 1
```

```
1: procedure DAgger(\pi_{\star}, \Phi^{\mathsf{IL0}}, T)
```

2:
$$\hat{\pi}_0 \leftarrow \text{Train}(\Phi^{\text{IL}0})$$
 \triangleright initial model, iff $\Phi^{\text{IL}0} = \Phi^{\text{OPT}}$

for $i \leftarrow 1$ to T do \triangleright at each imitation learning iteration 3:

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

5: Sample a
$$K$$
-solution using $\pi_i
ightharpoonup IL $(i, \hat{\pi}_{i-1}, \pi_{\star})$$

6:
$$\Phi^{\text{IL}i} = \{(s, \pi_{\star}(s))\} \triangleright \text{ visited by } \pi_i \text{ and actions by } \pi_{\star}$$
7: $\Phi^{\text{DA}i} \leftarrow \Phi^{\text{DA}i-1} \sqcup \Phi^{\text{IL}i} \qquad \triangleright \text{ aggregate datasets}$

7:
$$\Phi^{DA_I} \leftarrow \Phi^{DA_I-1} \cup \Phi^{IL_I}$$
 > aggregate datasets

8:
$$\hat{\pi}_{i+1} \leftarrow \text{Train}(\Phi^{\text{DA}I})$$
 \triangleright preference model

10: **return** best
$$\hat{\pi}_i$$
 on validation \triangleright best preference mode



ALIC

Helg

Introductio Problem

Subspace of instances

Feature spa

Algorithm space

Performanc space

instance spac Preference se

Prefere learning

Conclusion

```
Require: T \ge 1
```

```
1: procedure DAgger(\pi_{\star}, \Phi^{\text{IL0}}, T)
2: \hat{\pi}_0 \leftarrow \text{Train}(\Phi^{\text{IL0}}) \Rightarrow \text{ini}
```

2: $\hat{\pi}_0 \leftarrow \text{Train}(\Phi^{\text{IL}0})$ \triangleright initial model, iff $\Phi^{\text{IL}0} = \Phi^{\text{OPT}}$

3: **for** $i \leftarrow 1$ **to** T **do** \triangleright at each imitation learning iteration

4: Let
$$\pi_i = \beta_i \pi_* + (1 - \beta_i) \hat{\pi}_{i-1}$$

5: Sample a K-solution using $\pi_i
ightharpoonup IL<math>(i, \hat{\pi}_{i-1}, \pi_{\star})$

6: $\Phi^{\mathsf{IL}i} = \{(s, \pi_{\star}(s))\} \triangleright \text{ visited by } \pi_i \text{ and actions by } \pi_{\star}$

7: $\Phi^{DAi} \leftarrow \Phi^{DAi-1} \cup \Phi^{ILi}$ \triangleright aggregate datasets

8: $\hat{\pi}_{i+1} \leftarrow \mathsf{Train}(\Phi^{\mathsf{DA}i})$ \triangleright preference model

9: end for

10: **return** best $\hat{\pi}_i$ on validation \triangleright best preference model

11: end procedure



ALIC

Helg

Problem

Subspace of nstances

Algorithm

Performance space

instance spac Preference se

Preferen

onclusion

```
Require: T > 1
 1: procedure DAgger(\pi_{\star}, \Phi^{\text{IL0}}, T)
            \hat{\pi}_0 \leftarrow \text{Train}(\Phi^{\text{IL}0}) \triangleright initial model, iff \Phi^{\text{IL}0} = \Phi^{\text{OPT}}
 2:
            for i \leftarrow 1 to T do \triangleright at each imitation learning iteration
 3:
                  Let \pi_i = \beta_i \pi_* + (1 - \beta_i) \hat{\pi}_{i-1}
 4:
                  Sample a K-solution using \pi_i 
ightharpoonup IL(i, \hat{\pi}_{i-1}, \pi_{\star})
 5:
                  \Phi^{\mathsf{IL}i} = \{(s, \pi_{\star}(s))\} \triangleright \text{ visited by } \pi_i \text{ and actions by } \pi_{\star}
 6:
                  \Phi^{DAi} \leftarrow \Phi^{DAi-1} \sqcup \Phi^{ILi}
                                                                       ▷ aggregate datasets
 7:
                  \hat{\pi}_{i+1} \leftarrow \mathsf{Train}(\Phi^{\mathsf{DA}i})
                                                                           ▷ preference model
 8:
            end for
 9:
            return best \hat{\pi}_i on validation \triangleright best preference model
10:
11: end procedure
```



Deviation from optimality, /

ALIC

Helg

Introduction

Problem space

instances

r catare sp

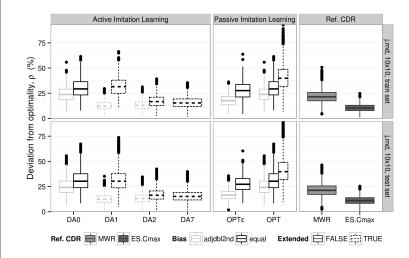
Algorithm

Performanc space

Footprints in instance space

Preference :

Conclusion





Framework for Algorithm Learning

ALIC

Helg

Introduction Problem

Subspace of instances

Feature spa

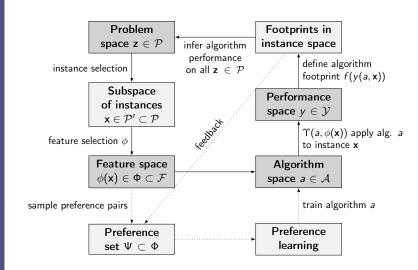
Algorithm

Performanco space

instance space

. . .

Conclusion





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
- \star Define features to grasp the essence of visited k-solutions
- * Success is highly dependent on the preference pairs
 - $\star \Psi_n$ reduces the preference set without loss of
 - * Stepwise bias is needed to balance time dependent



ALICE

Helg

ntroductio

Subspace of nstances

Algorithm space

Performance space Footprints in

nstance spac Preference se Preference learning 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.
- \star Define features to grasp the essence of visited k-solutions
- * Success is highly dependent on the preference pairs introduced to the system:
 - $\star \Psi_p$ 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



ALICE

Helg

ntroductio

Subspace of nstances

Feature spa Algorithm space

Performance space Footprints in nstance spac

Preference earning

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.
- \star Define features to grasp the essence of visited k-solutions
- * Success is highly dependent on the preference pairs introduced to the system:
 - $\star \Psi_p$ 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.



ALICE

Helg

Problem

Subspace of nstances

Feature spac Algorithm Space Performance

Footprints in nstance space Preference sel Preference earning 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.
- \star Define features to grasp the essence of visited k-solutions
- * Success is highly dependent on the preference pairs introduced to the system:
 - $\star \Psi_p$ 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.



ALICE

Helga

Problem
pace
Subspace of

nstances eature spac

Algorithm

pace
Performance

pace

Gootprints in

instance space

cootprints in instance space Preference set Preference earning Conclusions 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.
- \star Define features to grasp the essence of visited k-solutions
- * Success is highly dependent on the preference pairs introduced to the system:
 - $\star \Psi_p$ reduces the preference set without loss of performance.
 - \star 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



ALICE

Helga

Problem pace subspace of astances

eature spac algorithm pace erformance

cotprints in istance space Preference set 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.
- \star Define features to grasp the essence of visited k-solutions
- * Success is highly dependent on the preference pairs introduced to the system:
 - $\star \Psi_p$ 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



ALICE

Helga

roblem pace ubspace of istances

Algorithm

Performance

pace

Sootprints in

Instance space

Preference so

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.
- \star Define features to grasp the essence of visited k-solutions
- * Success is highly dependent on the preference pairs introduced to the system:
 - $\star \Psi_p$ reduces the preference set without loss of performance.
 - \star 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.



ALICE

Helg

Introduction

Subspace o

instances Feature sn:

Algorithm

Performanc space

Footprints in instance space

Preference

Conclusion

- \star 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.
- * In sequential decision making, all future observations are dependent on previous operations. 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.



ALICE

Helg

Introductio

Subspace of instances

Feature spa Algorithm space

Performance space

Preference : Preference learning

- \star 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.
- * In sequential decision making, all future observations are dependent on previous operations. 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.



ALICE

Helga

Introducti Problem space

Subspace of nstances

Algorithm pace Performance

instance spa Preference s Preference learning

- \star 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.
- * In sequential decision making, all future observations are dependent on previous operations. Active update procedure using DAgger ensures sample states the learned model is likely to encounter is integrated to Ψ^{DAi}_p.
- * Instead of reusing same problem instances, extend the training set with new instances for quicker convergence of DAgger.



ALICE

Helga

ntroductio Problem space

Subspace of nstances

Algorithm pace Performance

Footprints in instance space
Preference se
Preference
learning

- \star 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.
- * In sequential decision making, all future observations are dependent on previous operations. 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.



ALICE

Helga

Problem pace Subspace of instances

algorithm pace Performance pace

Footprints in Instance space Preference se Preference Earning

- \star 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.
- * In sequential decision making, all future observations are dependent on previous operations. 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.



Future work

ALIC

Helg

Introduction

Problem

Subspace of

Feature space

Algorithm

Performance

Footprints in instance space

Preference s

Preference

Camalantan

Main conclusions:

*



Thank you for your attention

ALIC

Helg

Problem space

Subspace of instances

Feature spa Algorithm space

Performanc space

Footprints in nstance space Preference sel

Conclusions

Helga Ingimundardóttir hei2@hi.is

Supplementary material:

- * Shiny application
- * Github.

