TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Is Deep Reinforcement Learning ready for solving industrial optimization problems?

Yingqian Zhang (TU/e)

ACP Summer School 2023, Leuven, 13 July 2023

What I cover today

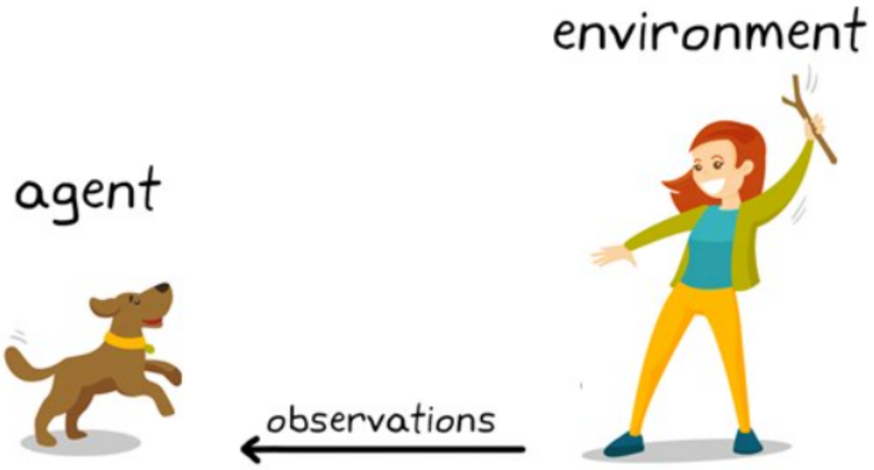
- A very brief introduction to DRL
- Example 1: learning to make online decisions (in collaborative picking)
- Example 2: (hybrid) learning to schedule jobs in manufacturing

TU/e

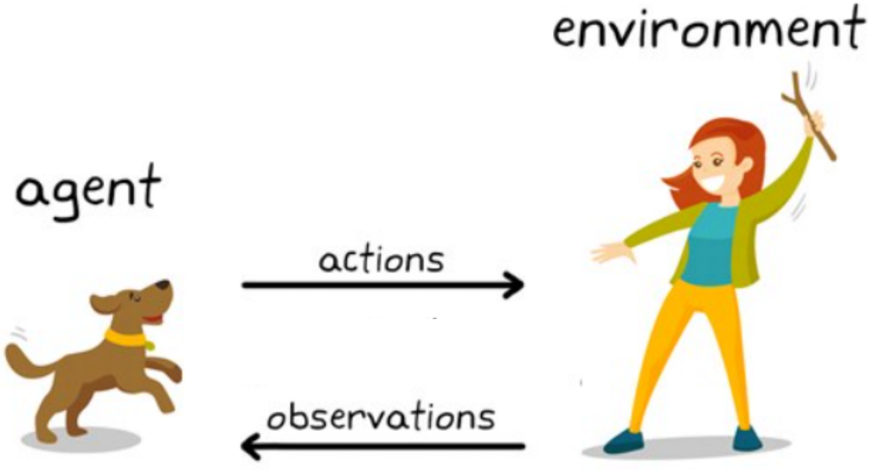
Intro to (deep) reinforcement learning

Credit:
Zaharah Bukhsh

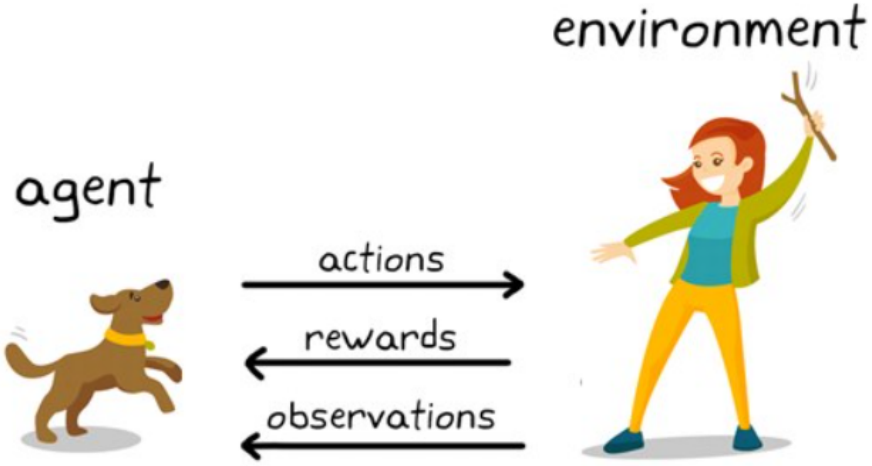
Credit:
Zaharah Bukhsh



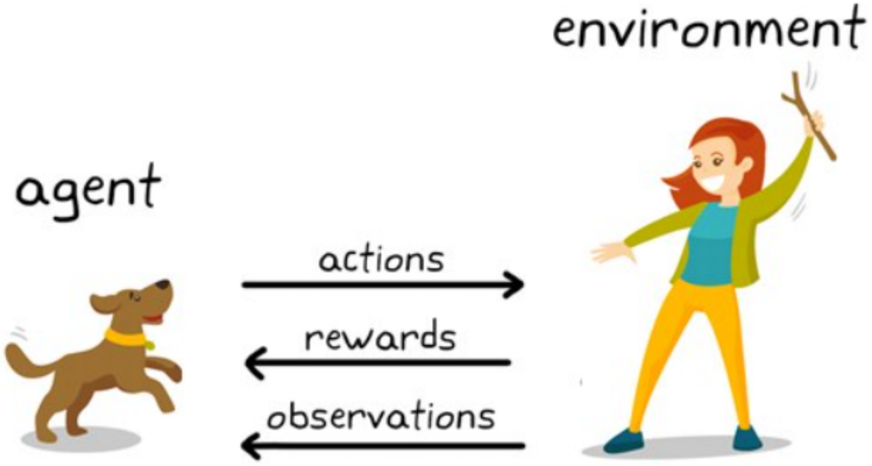
Credit:
Zaharah Bukhsh



Credit:
Zaharah Bukhsh



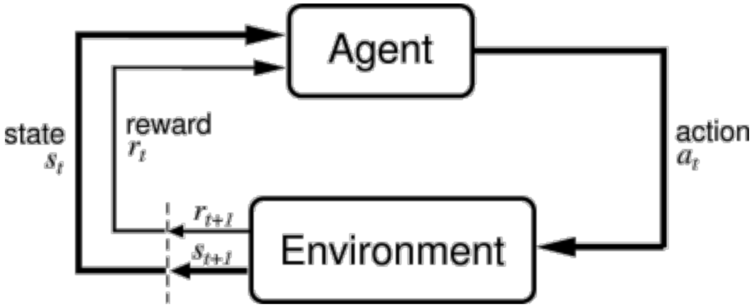
Credit:
Zaharah Bukhsh



- Reinforcement learning is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal.
- Trial-and-error search
- Delayed reward

Credit: Zaharah Bukhsh

Agent and Environment



Agent–environment interaction in a Markov decision process

At each step *t* the agent:

- Executes action A_t
- Receives observation S_t
- Receives scalar reward R_t

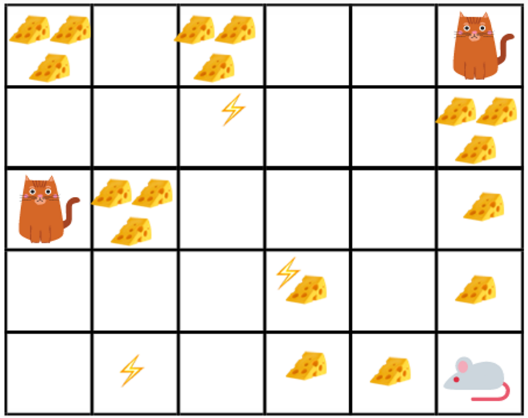
The environment:

- Receives action At
- Emits observation St+1
- Emits scalar reward Rt+1

t increments after interaction with the environment

Introduction to Reinforcement learning

Agent interaction with environment

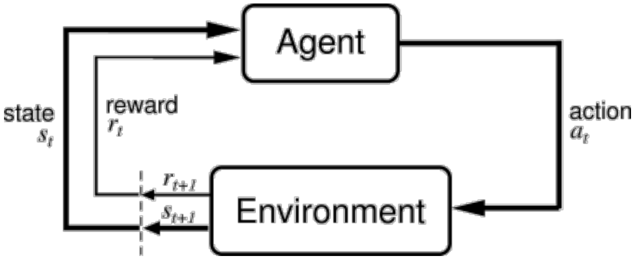


"RL is a computational approach to understanding and automating **goal-directed learning** and **decision making**. It is distinguished from other computational approaches by its emphasis on **learning** by an **agent** from direct **interaction with its environment**, **without requiring exemplary supervision** or complete models of the environment" (Sutton and Barto, 2015).

- Formalism for learning decision-making and control from experience
- Framework for learning to solve sequential decision-making problem

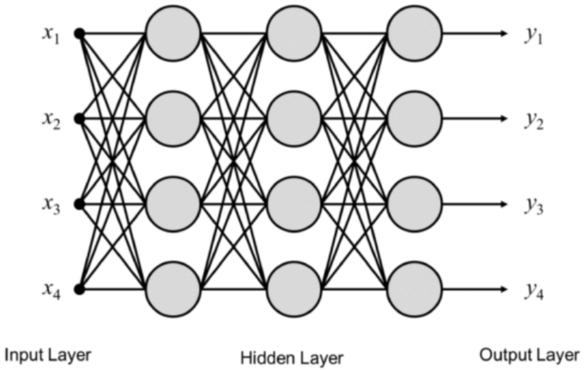
What is “deep” RL?

Reinforcement Learning



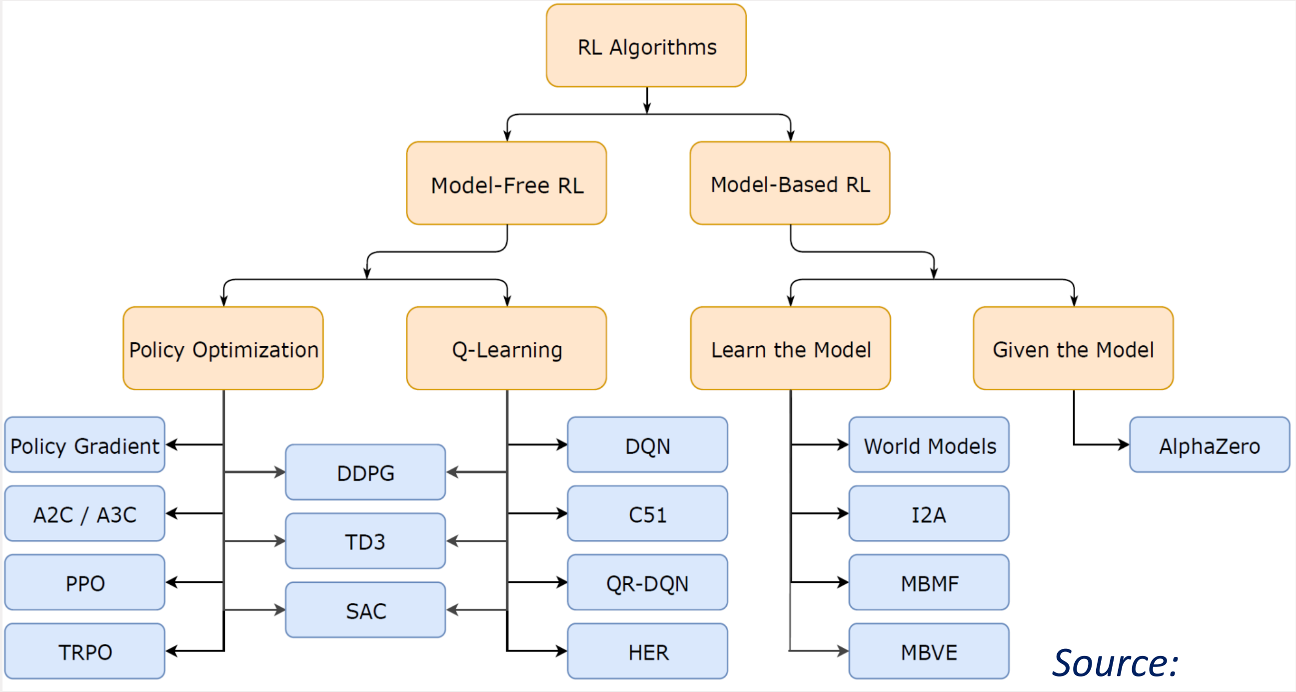
+

Deep Learning



Deep Reinforcement Learning

A non-exhaustive taxonomy of algorithms in modern RL



Source: spinningup.openai.com

Success stories of DRL

Human-level control through DRL (1/4)

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529-533.

- 49 Atari games
- From pixel to actions (no domain knowledge)
- The change in score is the reward.
- Same algorithm.
- Same function approximator w/3M free parameters.
- Same hyperparameters
- Roughly human-level performance on 29 out 49 games.

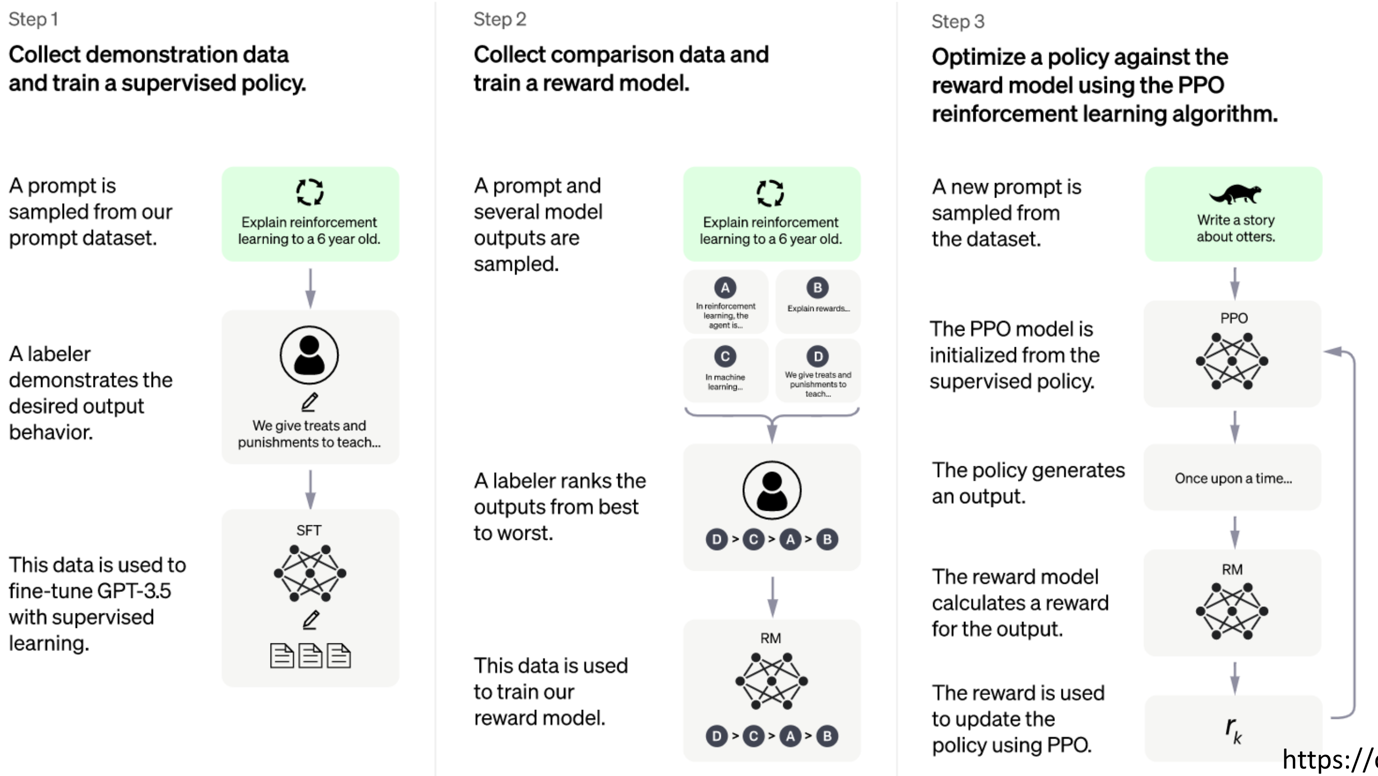
AlphaGo Zero (2/4)

AlphaGo beats Go master Lee Se-dol (12 March 2016)

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... & Hassabis, D. (2017). Mastering the game of go without human knowledge. *nature*, *550*(7676), 354-359.


to ChatGPT
AI account to c
OpenAI

ChatGPT: Optimizing Language Models for Dialogue – (3/4)



https://openai.com/blog/chatgpt/

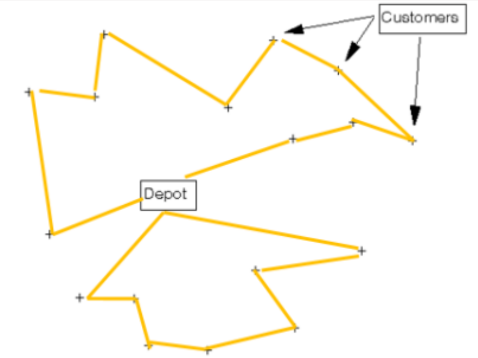
Solving combinatorial optimization problems (4/4?)

source: Wefabricate

source: NS

source: vanderlande

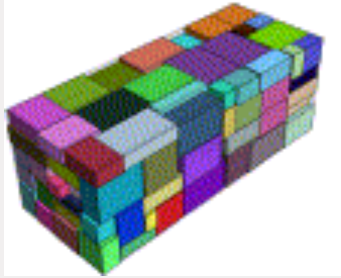
Real-world optimization problems



(Combinatorial) optimization problems:

with objectives, constraints

- Route planning for package delivery
  - Vehicle routing problem (VRP)
- Stacking/packing problems in harbours/warehouses; Train shunting
  - (3D) bin-packing problem
- Scheduling jobs on production lines
  - Job shop/machine scheduling problem



Real-world optimization problem

Two types of problems

1. Stochastic, sequential decision making problems: need quick, online decisions, e.g. ambulance dispatching, order batching in e-commerce warehouses
2. Full information available, although some executions could have randomness, e.g. (usually) package delivery, scheduling problems

Successful story 4.1

VANDERLANDE

End-to-end DRL for Collaborative picking

Igor Smit
Luca Begnardi

Order Picking

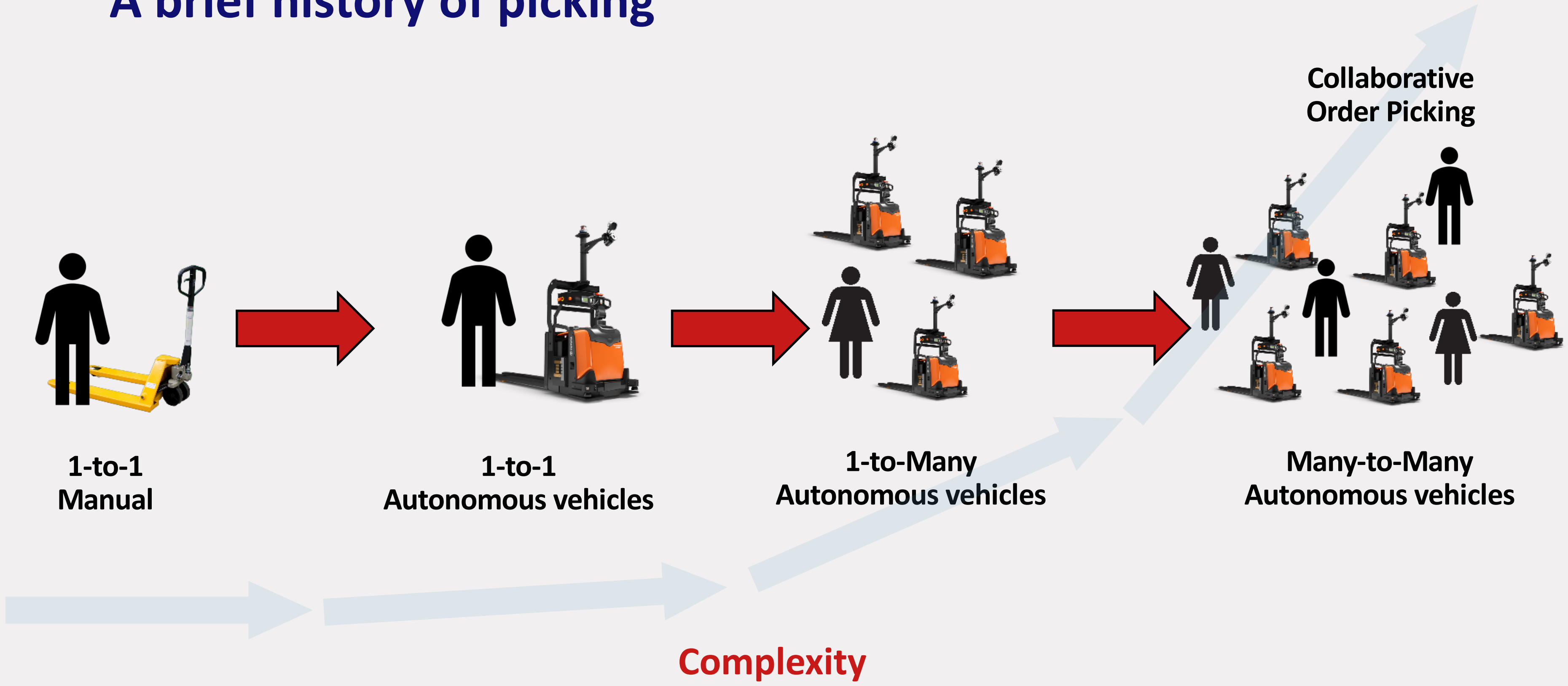
Order Picking is a crucial component of warehouse operation

- Order batching
- Order releasing
- **Picker routing:** which retrieving tasks should be assigned to each picker in order to optimize a certain metric, such as total order tardiness

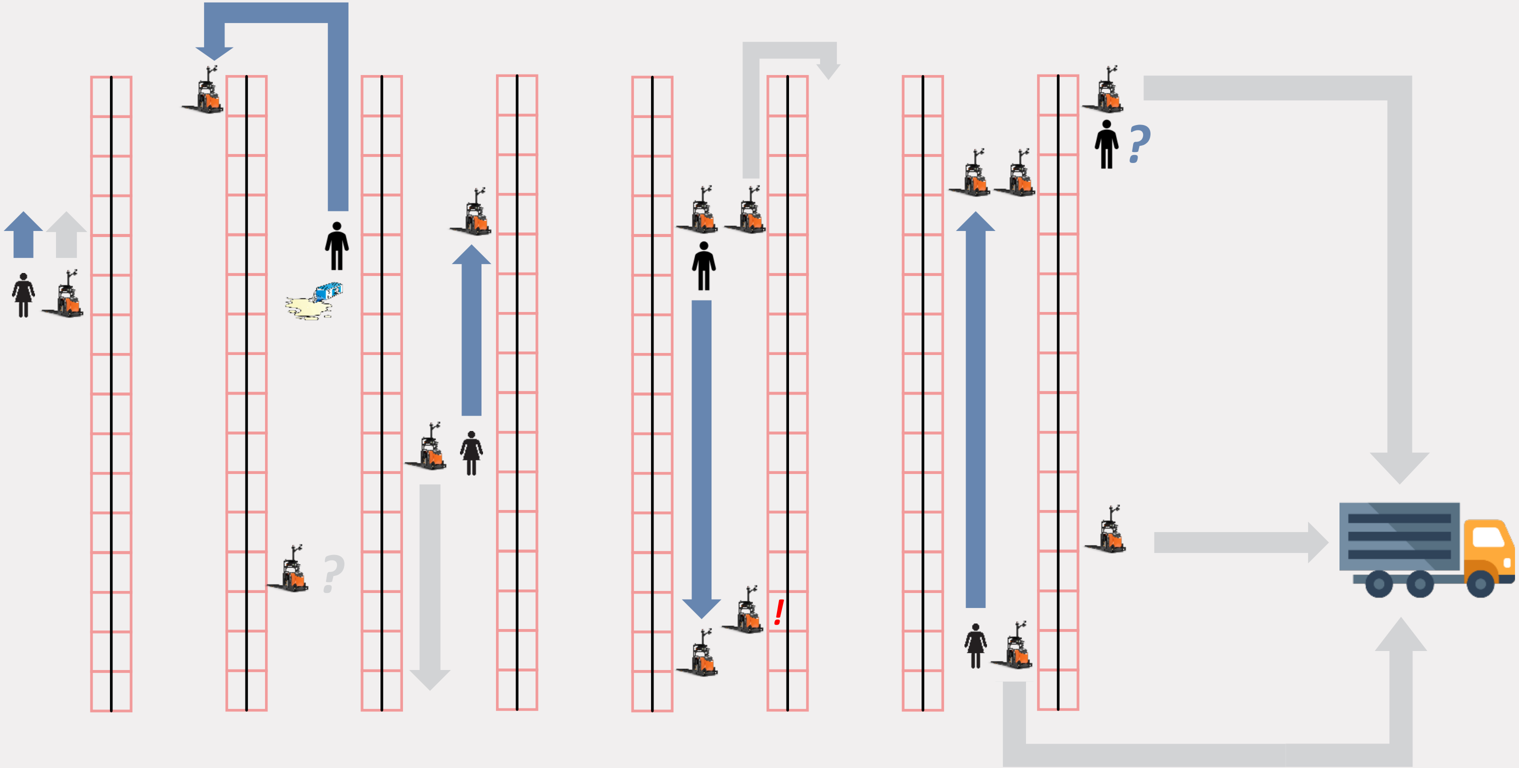
Traditional human-only order picking

- Humans handle all the work
- Highly inefficient: only 30% of time spent picking, on average.

A brief history of picking



Orchestrating chaos



Collaborative Order Picking: Challenge

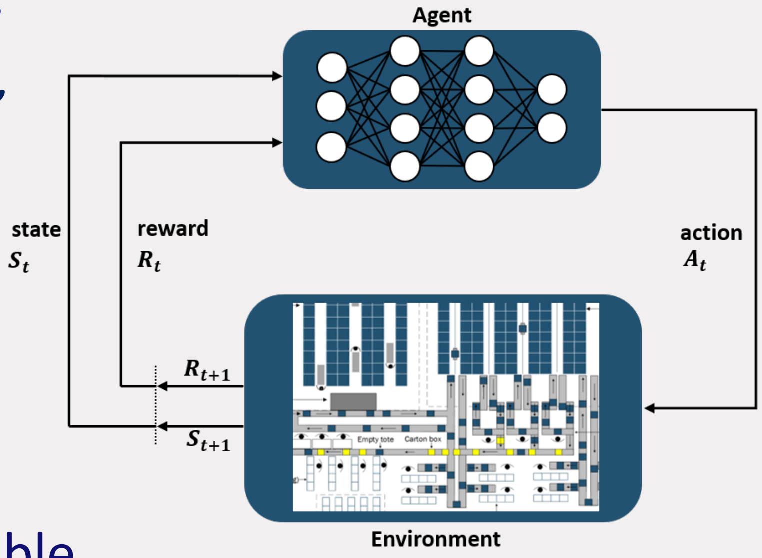
How to allocate pickers to AMRs (autonomous mobile robots)?

Solution requirements

- Online allocation/decision
- Handling uncertainty and congestion
- Handling different picker/AMR numbers, different warehouse sizes

Optimization problem

- Develop a 'picker optimizer' in human-robot collaborative picking
- *Allocating human pickers to AMRs, such that the total picking time is minimized (i.e., max pick rate)*



What we know

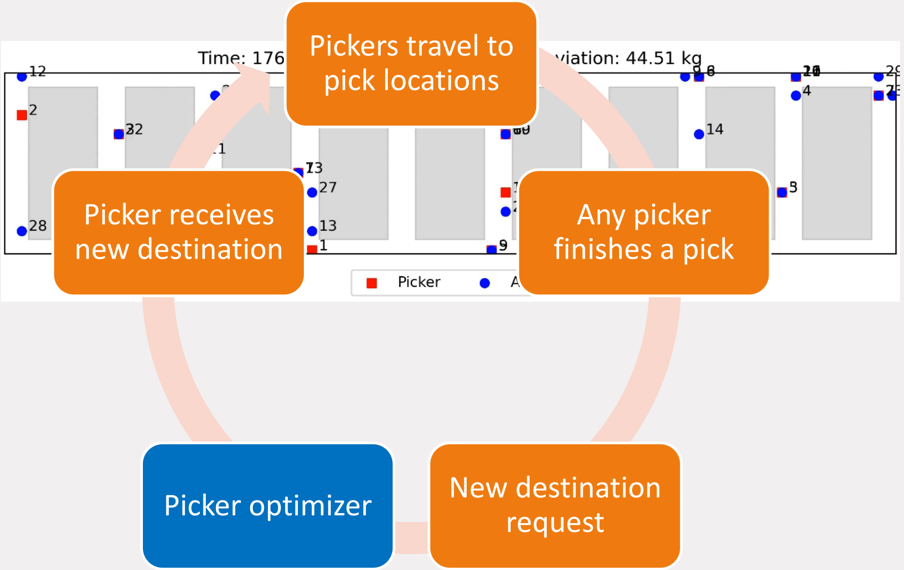
- Business rules are available
- Data (orders, locations of items) available

Environment: simulation model

Representing interactive processes between pickers, AMRs, and the picker optimizer.

Uncertainties

- stochastic picker/AMR speeds and picking times
- picking disruptions
- congestion causing delays for the AMRs related to overtaking procedures

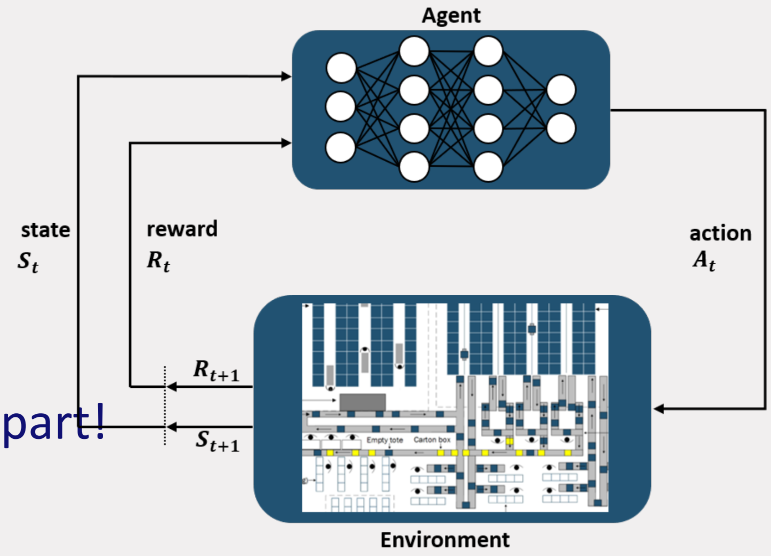


DRL approach

We have built the environment

Next:

- MDP (Markov decision process) formulation
  - State space (and embedding): most tricky part!
  - Action space
  - Reward function



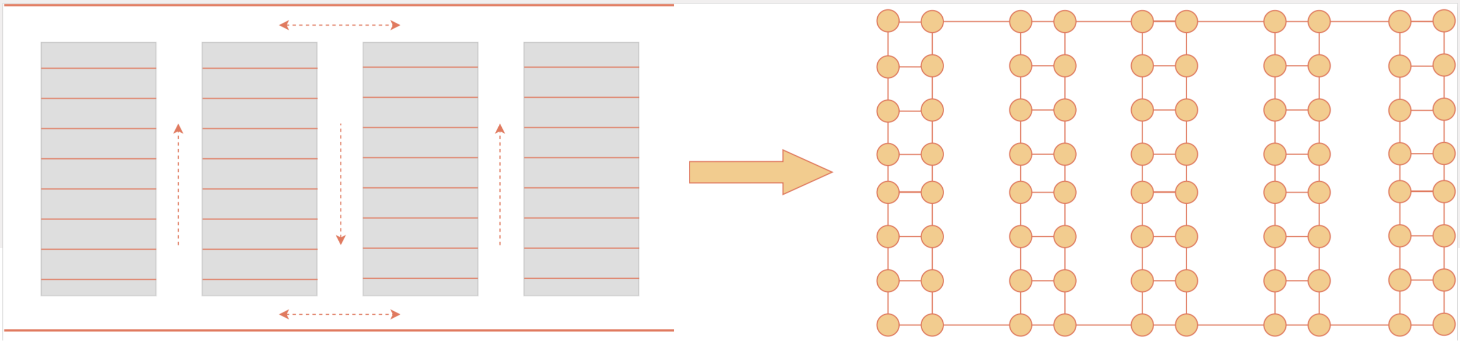
Spatial input representation

Classical feed-forward neural networks not suitable as

- Cannot adapt to different problem instances due to fixed input size; hard to capture important spatial information

Instance as a graph

- Nodes: locations in warehouse; Edges: how to travel between locations



State Representation

Node features capture information for each node/location

1. Controlled picker
2. AMR information
3. Picker information
4. Node location information
5. Node neighborhood information

Reward Function

Action Space:

- New picker destination
- Truncated action space: current or next AMR destination where no other picker is going

- Reward function: Penalty on time that passes

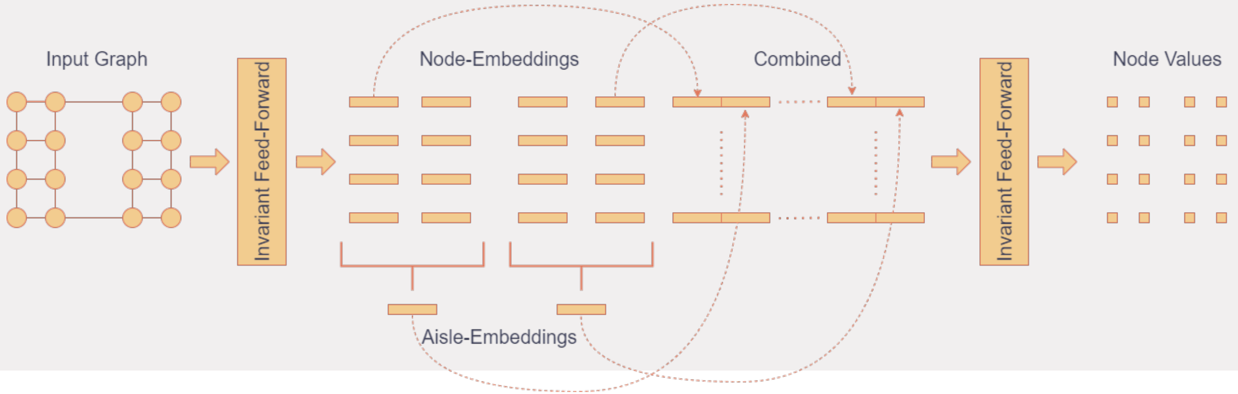
$$R_t^{\text{efficiency}} = \tau_{t-1} - \tau_t$$

Agent architecture design

Network must handle graph space and extract spatial information

~~Regular graph neural networks → Message passing~~

We adapt Invariant Feed-Forward network (Alomrani et al., 2022), with aisle-embedding: capture spatial relations in warehouse, size agnostic



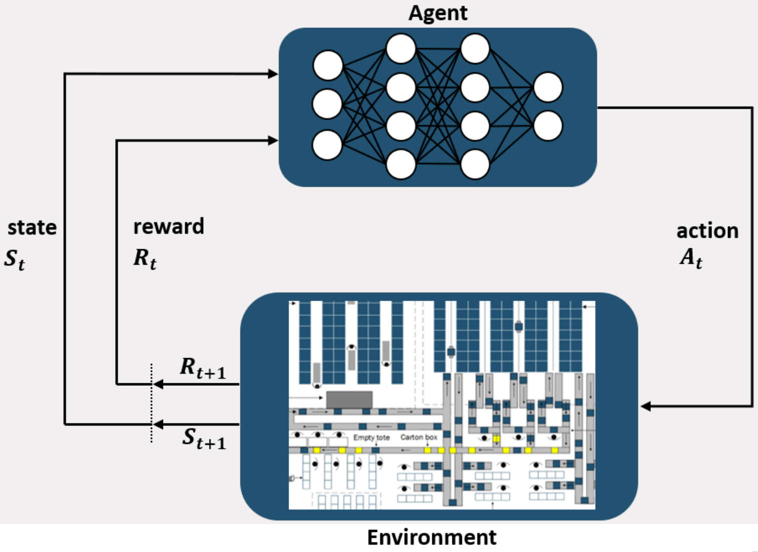
Learning Algorithm

Proximal Policy Optimization (PPO) (Schulman et al., 2017)

- Policy-based method: policy network directly outputs the actions

Actor-critic

- Actor network: suggests actions
- Critic network: estimates the advantage function (how good is the selected action)



Experiments

Experimental settings

- Performance of fixed warehouse sizes: picking time
- Picker/AMR transferability
- Warehouse size transferability

Multiple warehouse types

Warehouse Type	Aisles	Aisle Depth	Locations	Pickers	AMRs	Picks
XS	7	7	98	4	7	≤ 100
S	10	10	200	10	25	5000
M	15	15	450	20	50	7500
L	25	25	1250	30	90	7500
XL	35	40	2800	60	180	15000

Benchmark algorithms

- VI benchmark: rule-based method considers the distance of potential picks and also tries to spread the pickers across aisles
- Greedy : assign a picker to the closest available location where an AMR is going, and no other picker is already going.
- MILP: assuming no uncertainties/randomness, no congestions, in XS environment

  min C , subject to

$$\sum_{k\in\mathcal{K}} A_{i,k} = 1 \qquad \forall i \in \mathcal{N} \tag{2.3}$$

$$A_{i,k} - A_{i',k} \leq 1 - (U_{i,i'} + U_{i',i}) \qquad \forall i, i' \in \mathcal{N}, i \neq i', k \in \mathcal{K} \tag{2.4}$$

$$A_{i,k} + A_{i',k} \leq 1 + (U_{i,i'} + U_{i',i}) \qquad \forall i, i' \in \mathcal{N}, i \neq i', k \in \mathcal{K} \tag{2.5}$$

$$B^K_{i.k} \geq \tau^{o,K}_{k,i} - M \cdot (1 - A_{i,k}) \qquad \forall i \in \mathcal{N}, k \in \mathcal{K} \tag{2.6}$$

$$\sum_{k\in\mathcal{K}} B^K_{i.k} \geq \sum_{k\in\mathcal{K}} F^K_{i',k} + \tau^K_{i',i} \cdot U_{i',i} - M \cdot (1 - U_{i',i}) \qquad \forall i, i' \in \mathcal{N}, i \neq i' \tag{2.7}$$

$$F^K_{i,k} \geq F^R_{i,r} - M \cdot \left(2 - A_{i,k} - a^R_{i,r}\right) \qquad \forall i \in \mathcal{N}, k \in \mathcal{K}, r \in \mathcal{R} \tag{2.8}$$

$$\sum_{r\in\mathcal{R}} B^R_{i,r} \geq \left(\sum_{r\in\mathcal{R}} F^R_{i',r} + \tau^R_{i',i}\right) \cdot u^R_{i',i} \qquad \forall i, i' \in \mathcal{N}, i \neq i' \tag{2.9}$$

$$B^R_{i,r} \geq \tau^{o,R}_{r,i} \cdot a^R_{i,r} \qquad \forall i \in \mathcal{N}, r \in \mathcal{R} \tag{2.10}$$

$$B^R_{i,r} \geq B^K_{i,k} - M \cdot (2 - A_{i,k} - a^R_{i,r}) \qquad \forall i \in \mathcal{N}, r \in \mathcal{R}, k \in \mathcal{K} \tag{2.11}$$

$$F^R_{i,r} = B^R_{i,r} + \eta^L_i \cdot a^R_{i,r} \qquad \forall i \in \mathcal{N}, r \in \mathcal{R}, \tag{2.12}$$

DO NOT READ!

(is incomplete)

Experiment 1: performance on deterministic, XS instances

(a) Instances of type XS with diverse starting.

Instance	DRL	Greedy	VI Benchmark	MILP	MILP gap (%)
1	154	154	355	**149**	17.8
2	**187**	190	397	**187**	6.0
3	155	167	299	**149**	12.2
4	**206**	248	269	212	17.5
5	227	236	277	**206**	15.9

(b) Instances of type XS without diverse starting.

Instance	DRL	Greedy	VI Benchmark	MILP	MILP gap (%)
1	**244**	262	355	**244**	28.2
2	**249**	253	297	271	28.1
3	**265**	272	299	267	29.3
4	**240**	257	269	245	22.8
5	**251**	255	277	260	30.9

Compared to MILP solver: *small instances without randomness and overtaking penalty. Each AMR does 1 pickrun.*

Gurobi: gaps *after 20 hours*

(DRL: solves in milliseconds)

TU/e

Experiment 1: performance with various sizes

Values indicate the average picking time over 100 evaluation episodes

	DRL		Greedy		VI Benchmark
Warehouse	Picking Time	%	Picking Time	%	Picking Time
S	**8586 ± 62**	**14.9**	10619 ± 59	−5.3	10087 ± 58
M	**8425 ± 46**	**21.0**	11023 ± 58	−3.3	10669 ± 41
L	**6540 ± 37**	**31.7**	9823 ± 33	−2.7	9569 ± 61
XL	**9010 ± 21**	**33.6**	13972 ± 44	−3.0	13570 ± 72

Experiment 2: Picker/AMR Transferability

(a) Warehouse type S.

	DRL		Greedy		VI Benchmark
Pickers/AMRs	Picking Time	%	Picking Time	%	Picking Time
7/15	**12825 ± 83**	**17.1**	15166 ± 74	2.0	15472 ± 87
10/20	**9206 ± 51**	**19.4**	11274 ± 69	1.3	11420 ± 56
10/30	**8221 ± 54**	**13.0**	10283 ± 60	−8.8	9447 ± 52
15/25	**6737 ± 42**	**21.5**	7994 ± 40	6.9	8583 ± 36
15/30	**5930 ± 34**	**24.7**	7804 ± 55	1.0	7879 ± 46
15/35	**5938 ± 35**	**16.6**	7550 ± 44	−6.0	7121 ± 38

(d) Warehouse type XL.

	DRL		Greedy		VI Benchmark
Pickers/AMRs	Picking Time	%	Picking Time	%	Picking Time
50/120	**12028 ± 23**	**40.2**	16816 ± 32	16.5	20142 ± 112
60/140	**10150 ± 20**	**40.7**	14312 ± 27	16.4	17118 ± 101
60/200	**9009 ± 44**	**35.6**	14293 ± 88	−2.2	13979 ± 87
80/180	**8106 ± 77**	**38.9**	11343 ± 30	14.6	13275 ± 83
80/200	**8011 ± 59**	**36.8**	11765 ± 52	6.4	12571 ± 91
80/220	**6947 ± 19**	**41.5**	10799 ± 40	9.1	11877 ± 84

Trained with:
- 10/25 for S
- 60/180 for XL

The trained policies are applied to scenarios with different numbers of pickers and AMRs and their ratios

Experiment 3: Warehouse Size Transferability

- Trained policies on specific sized are tested on different sized instances

Warehouse	Policy S	Policy M	Policy L	Policy XL	Greedy	VI Benchmark
S	$\mathbf{8586 \pm 62}$	9190 ± 53	8875 ± 58	8986 ± 51	10619 ± 59	10087 ± 58
M	$\mathbf{7931 \pm 42}$	8425 ± 46	8064 ± 41	8220 ± 37	11023 ± 58	10669 ± 41
L	6877 ± 31	7190 ± 42	$\mathbf{6540 \pm 37}$	6877 ± 23	9823 ± 33	9569 ± 61
XL	9478 ± 20	11275 ± 33	$\mathbf{8567 \pm 24}$	9010 ± 21	13972 ± 44	13570 ± 72

TU/e

Ablation study: architecture comparison

INV-FF: invariant feed-forward network without aisle embedding

GIN: Graph Isomorphism Network

GCN: Graph Convolutional Network

Warehouse	INV-FF	AISLE-EMB	GIN	GCN
S	8689 ± 58	$\mathbf{8586 \pm 62}$	8869 ± 55	11677 ± 67
M	8628 ± 40	$\mathbf{8425 \pm 46}$	14151 ± 75	13851 ± 65
L	6602 ± 29	$\mathbf{6540 \pm 37}$	11723 ± 76	14419 ± 88

Picking performance

Warehouse	INV-FF	AISLE-EMB	GIN	GCN
S	$\mathbf{1.44 \pm 0.03}$	2.16 ± 0.03	3.10 ± 0.03	6.02 ± 0.05
M	$\mathbf{1.53 \pm 0.03}$	2.22 ± 0.03	3.25 ± 0.03	6.28 ± 0.05
L	$\mathbf{1.53 \pm 0.03}$	2.41 ± 0.03	3.41 ± 0.04	6.65 ± 0.05
XL	$\mathbf{1.73 \pm 0.03}$	2.57 ± 0.04	3.77 ± 0.04	7.19 ± 0.06

Inference time (in milliseconds)

Humans are not robots

where is fairness?

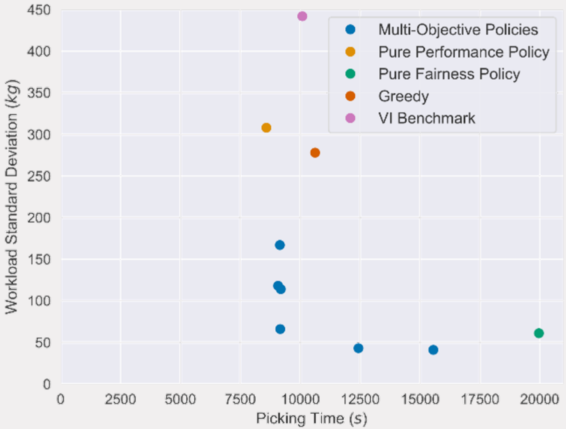
Multi-objective DRL

- Fairness: Minimize standard deviation of carried product masses
- Add to state space workload fairness features
  - Node-specific information, distributional information
- Add to reward function
  - penalty on increase in standard deviation
- Learning algorithm, adapted from the prediction-guided MORL (Xu et al., 2020)
  - A meta-policy approach, to present a non-dominated set showing the trade-offs

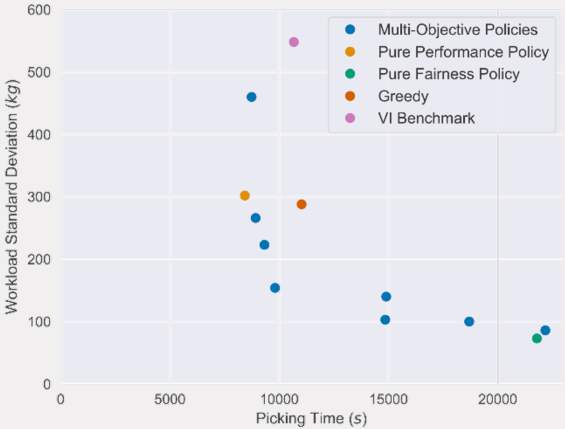
TU/e

Experiment: multi-objective fixed warehouse sizes

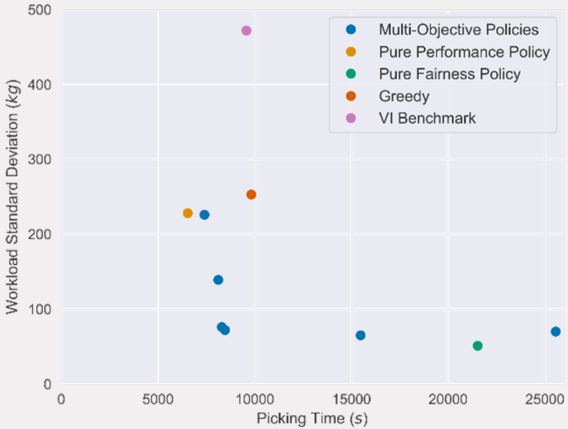
Warehouse S



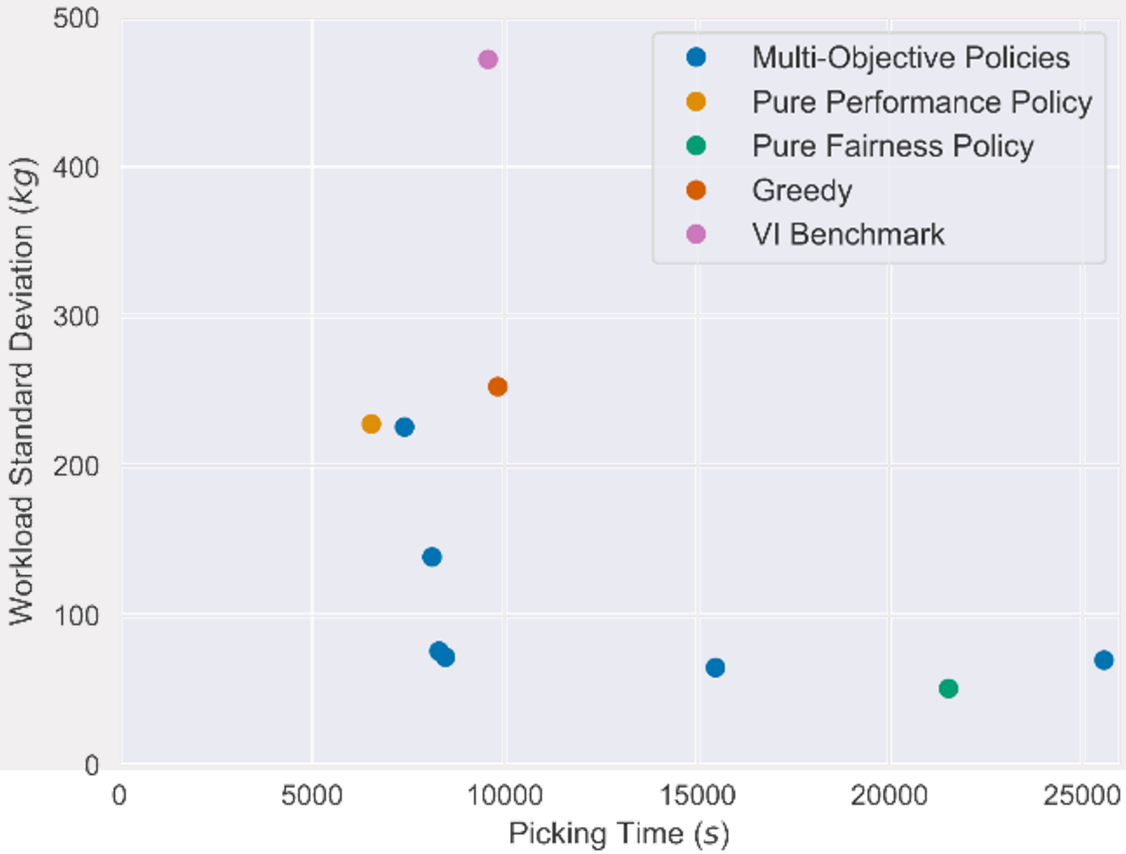
Warehouse M



Warehouse L



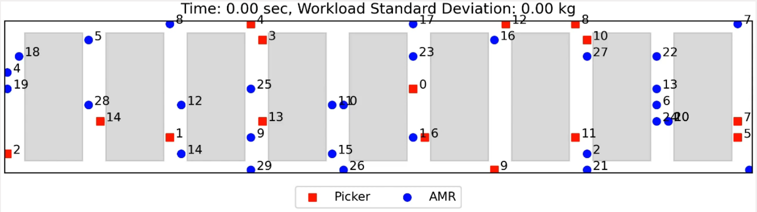
Experiment: multi-objective fixed warehouse sizes

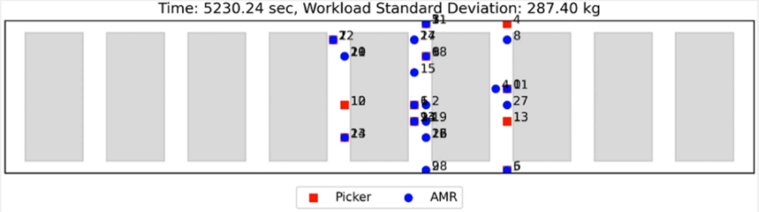


Warehouse L

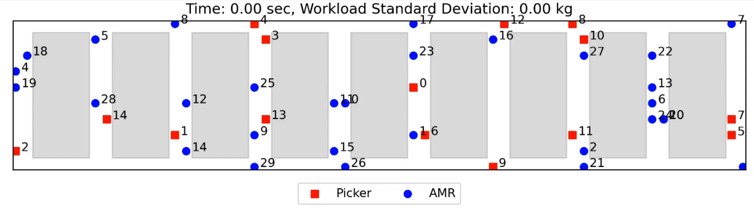
Inspecting Policy Behavior

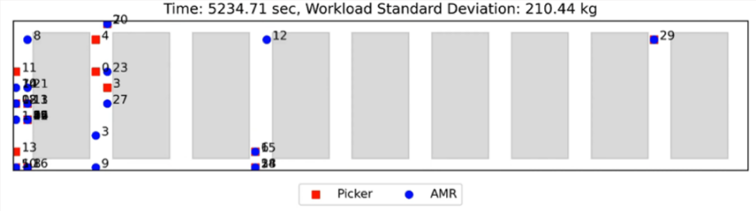
VI Benchmark



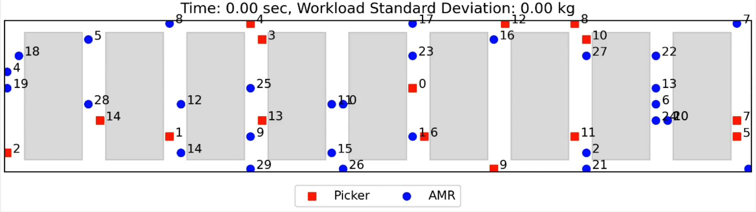


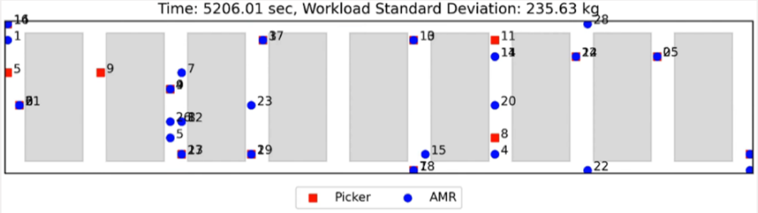
Greedy



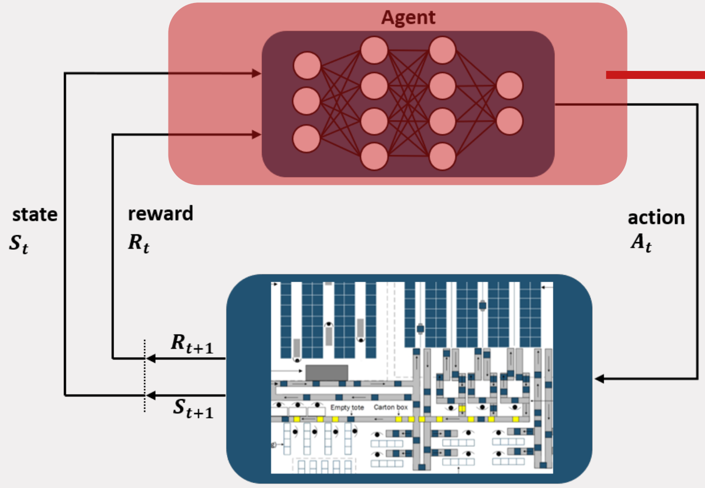


DRL

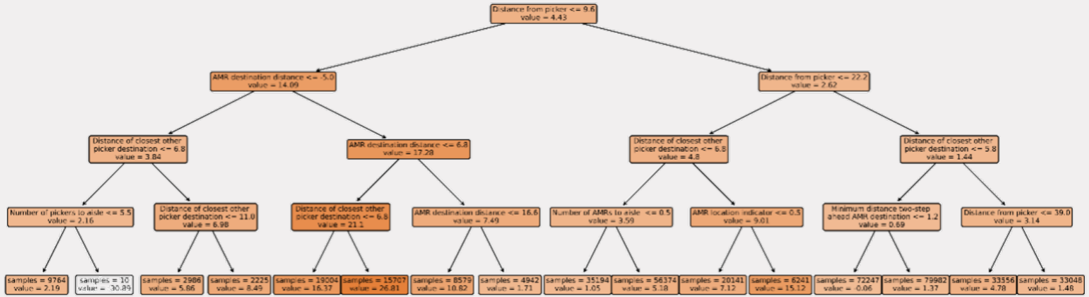




Policy analysis



Approximating policy behavior with a decision tree



Performance of DT

Approximating policy behaviour with a decision tree

Policy	Tree-Best PT	Tree-Best WF	DRL PT	DRL WF
1	16373 ± 139	41 ± 5	15555 ± 125	41 ± 4
2	12347 ± 83	41 ± 4	12431 ± 86	43 ± 4
3	9350 ± 65	81 ± 5	9164 ± 60	66 ± 4
4	9505 ± 62	161 ± 10	9188 ± 55	114 ± 8
5	9561 ± 77	197 ± 14	9074 ± 60	118 ± 7
6	9480 ± 79	225 ± 12	9149 ± 68	167 ± 9
Pure Performance	8785 ± 57	304 ± 21	8586 ± 62	308 ± 17
Pure Fairness	19141 ± 110	173 ± 20	19962 ± 86	61 ± 9

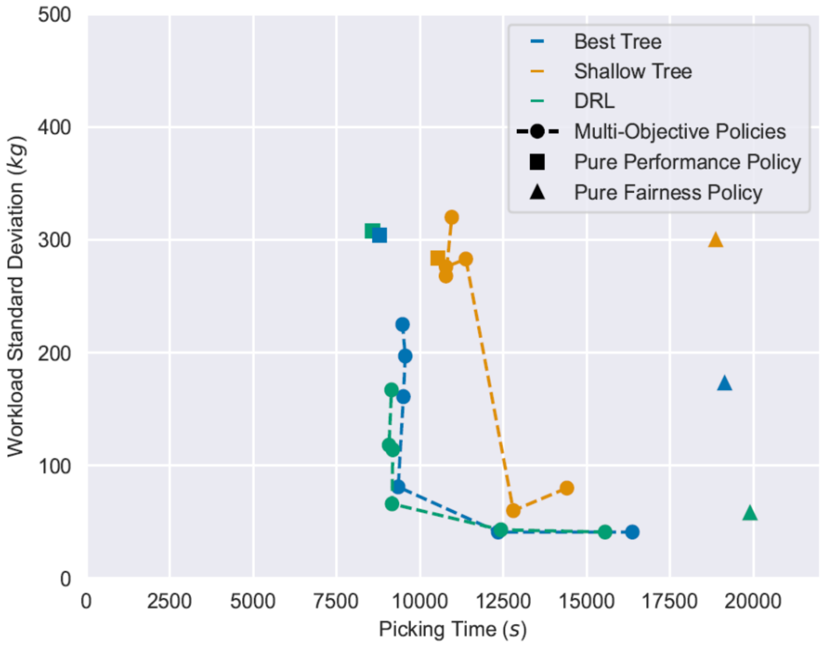


Figure 8.5: Performance evaluation of decision tree policies.

TU/e

Conclusions

- Efficiency improvements +- 40%
  - Higher warehouse capacity
  - Lower picking costs
- Good trade-off between efficiency and fairness
  - Explicitly outline achievable trade-offs
  - Simultaneous improvement of picking times and workload fairness
  - For large warehouse policy with 23.6% efficiency and 92% fairness improvements
- Good transferability

Is DRL ready for dynamic, real-world sequential decision making problems?

Yes! It gives better performance than handcrafted heuristic rules, which also generalizes well

Recipe

- Build a good (discrete event) simulation model
- Representing problem instances well:
  good features, (size-agnostic, general) architecture
- Applying model-free/model based DRL algorithms

Is DRL ready for dynamic, real-world sequential decision making problems?

Other real-world dynamic sequential decision problems

- Travelling repairwomen problem: Da Costa et al., Policies for the dynamic traveling maintainer problem with alerts, European Journal of Operational Research, 2023
- Order batching problem: Beeks et al., Deep Reinforcement Learning for a Multi-Objective Online Order Batching Problem, ICAPS 2022
- Cals et al., Solving the online batching problem using deep reinforcement learning, CAIE 2021
- Train shunting problem: Peer et al., Shunting Trains with Deep Reinforcement Learning, IEEE SMC, 2018
- Dynamic pricing in ad network: An Automated Deep Reinforcement Learning Pipeline for Dynamic Pricing, IEEE TAI, 2022

Is DRL ready for dynamic, real-world sequential decision making problems?

It can be even better…

Challenges & opportunities

- autoRL (Afshar et al. 2022)
- Instance representation (Ya et al. 2023)
- Non-Deep RL (Vos & Verwer, 2023)
- Simulation to real-world
- Adoption

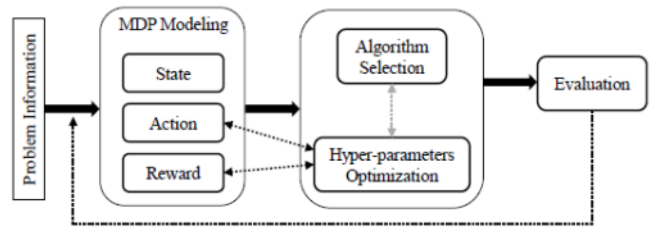


Fig. 1: Overview of the proposed automated DRL pipeline.

Afshar et al.,2022

Successful story 4.2

Flexible Job-Shop Scheduling

WEFABRICATE

Kjell van Straaten

Robbert Reijnen

Flexible Job Shop Scheduling Problem (FJSP)

- A set of n jobs, m machines
- Each job j contains an ordered sequence of operations $O_{i,j}$
- Each $O_{i,j}$ must be performed by one of the machines compatible with $O_{i,j}$
- $O_{i,j+1}$ can only start after $O_{i,j}$ is completed.
- $O_{i,j}$ has processing time on specific machine m
- Each machine can only process one operation at a time

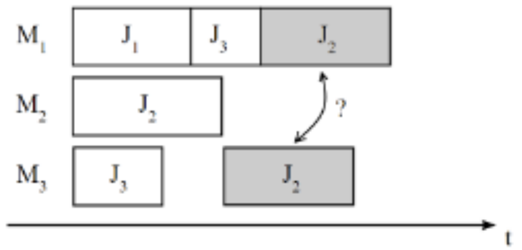


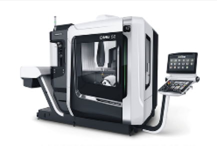
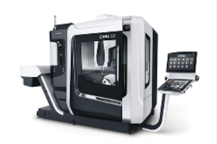
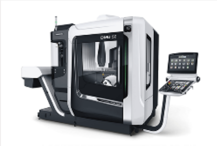
Figure 3.2: Flexible Job-Shop Scheduling

Objective: find a schedule that minimizes makespan
(the time when all jobs have been processed)

FJSP in practice…

Jobs (i.e., orders)
- Quantity
- Deadline
- Material availability
- Operations (milling steps)
    - Duration
    - Setup
- …

+


Machines
- Eligibility
- Tools
- Maintenance
- Operator
- …

?

->

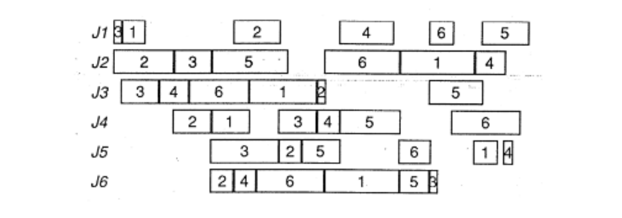


Figure 3.1: Job-Shop Scheduling, adopted from Yamada and Nakano (1992)

Objectives
- Makespan
- Operational Cost
    - Manual labor
    - Consumed Tools
    - Logistic movements
    - …

Existing approaches for FJSP

- Mathematical optimization models; Constraint programming
- Dispatching rules: (e.g., shortest processing times (SPT))
- Heuristics: constructive, metaheuristics
- Reinforcement learning based approaches
  - End-to-end DRL (Song et al., 2022)
  - Hybrid approaches: parameter controls of evolutionary algorithms with DRL (Chen et al., 2020)

What is missing in the literature?

- A comparison study between e2e DRL and a hybrid approach

Criteria	Algorithm
Performance	?
Runtime	?
Scalability	?
Robustness/generalization	?

on benchmark FJSP instances and a real-world instance

Two approaches

1. Ours: Self-Learning Effective Genetic Algorithm (SLEGA)
2. End-to-End DRL, adapted from (Song et al. 2022)

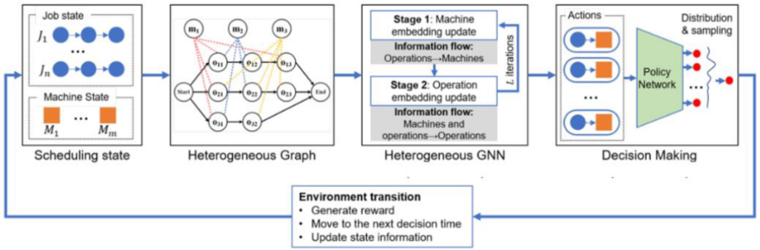


Figure 4.8: E2E-DRL Framework, adopted from Song et al. (2022)

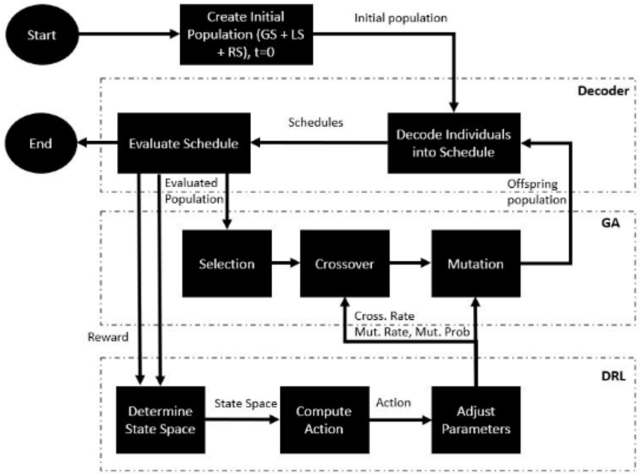


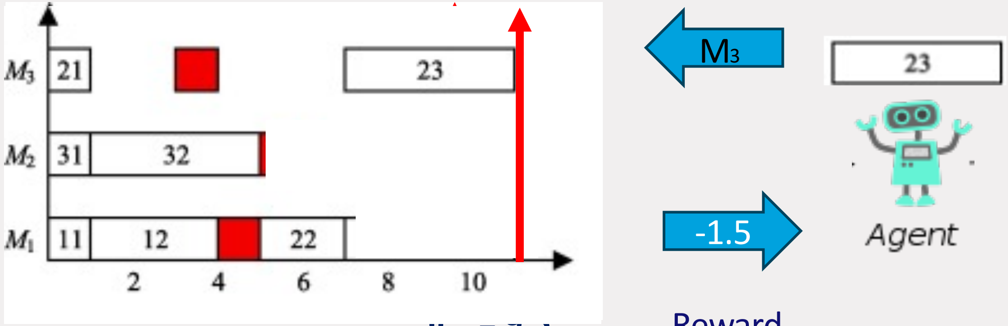
Figure 4.1: SLEGA framework

End-to-end DRL

E2E-DRL approach

Environment

Action



Reward

Tc* = 11

*Estimated Makespan

E2E-DRL

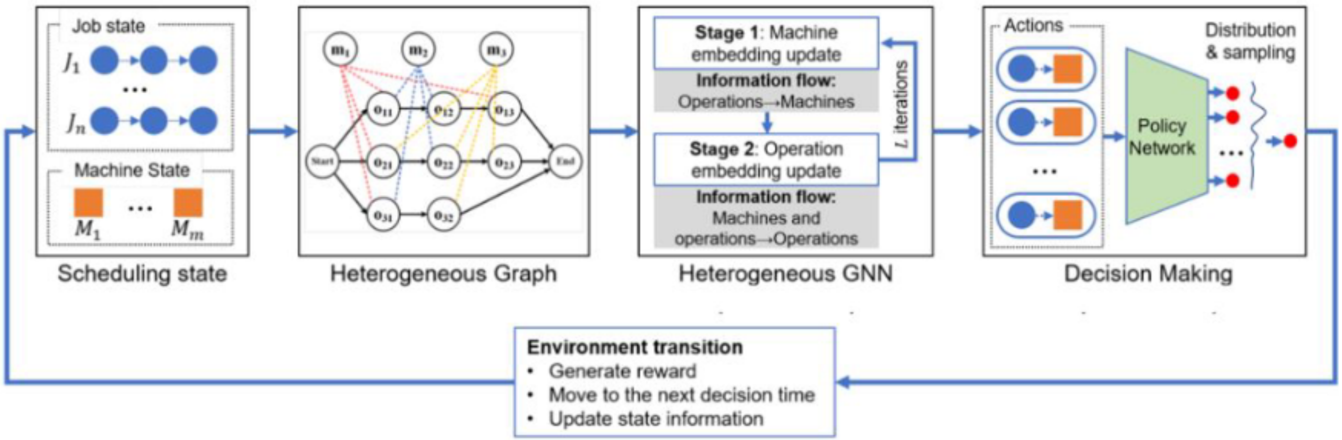


Figure 4.8: E2E-DRL Framework, adopted from Song et al. (2022)

E2E-DRL: MDP formulation (adapted from Song et al. 2022)

Actions:

Select operation to machine allocation action, from set of eligible actions.

Reward:

Absolute penalty if expected makespan increases.

E2E-DRL: state representation

State space: represented as a heterogeneous graph

- Operation nodes:
  Status; #neighboring machine; Average or actual processing time; #unscheduled jobs; Potential or actual start time; Time until the release date…
- Machine nodes:
  #neighboring operations; available time; utilization…
- Arcs:
  processing time; sequence dependent setup time…

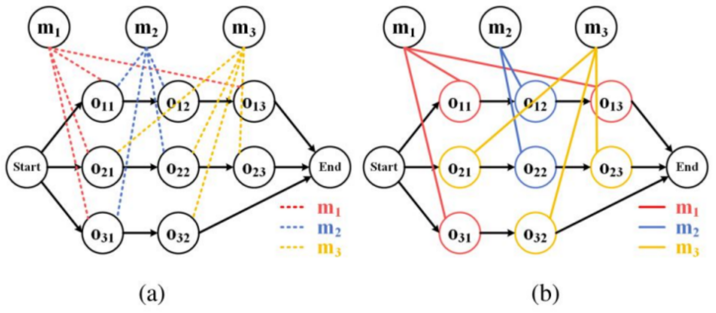
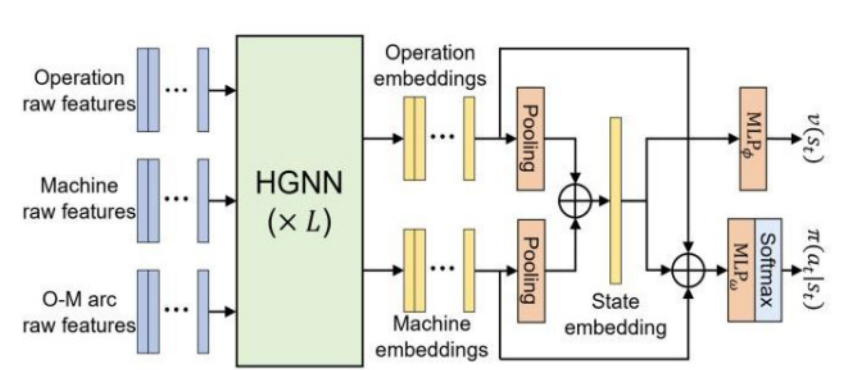


Figure 4.15: Heterogeneous Graph representation of FJSP. A dotted line means processable, while a solid line means scheduled. Adopted from Song et al. (2022)

State embedding

- GAT (graph attention network) is used for machine node embedding
- The message-passing step: a machine node aggregates information from only the direct neighbors (eligible operations on that machine).
- The operation node is then embedded given the eligible machines, the previous node, the next node and the node itself.

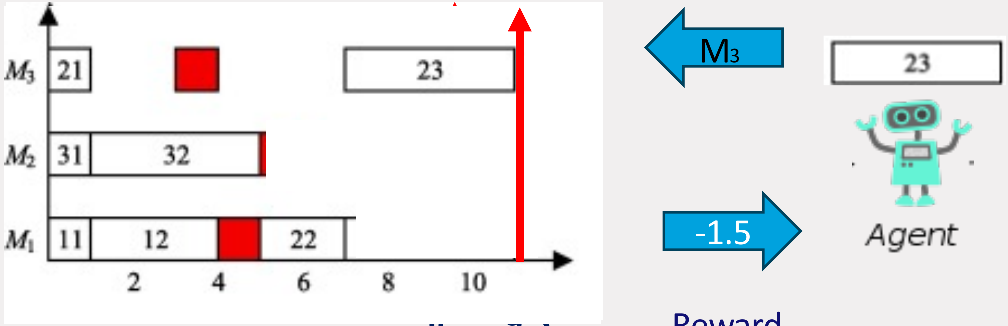
PPO with actor-critic structure



E2E-DRL approach

Environment

Action



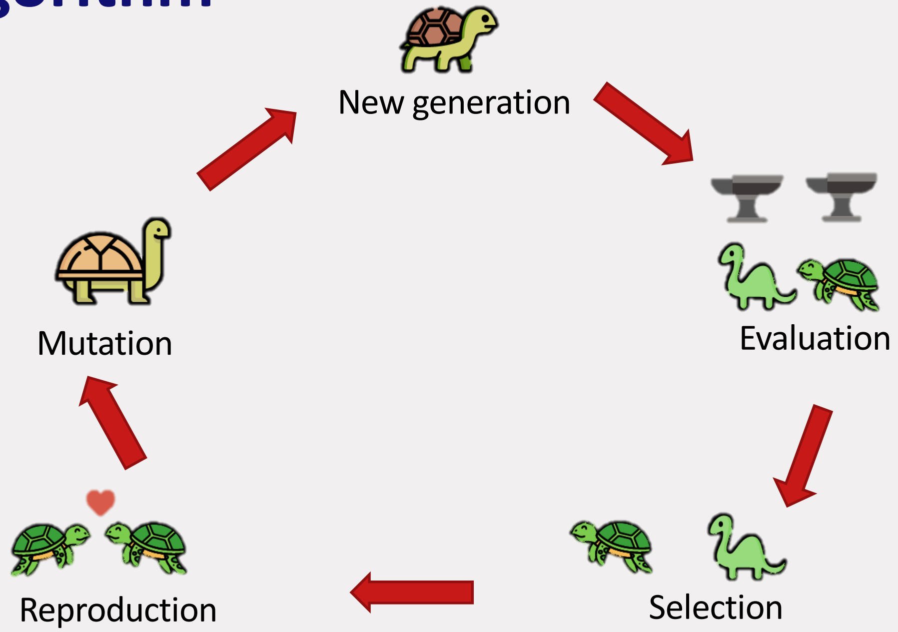
Reward

Tc* = 11

*Estimated Makespan

A hybrid GA-DRL approach

Genetic algorithm



Credit: Robbert Reijnen based on https://www.generativedesign.org/02-deeper-dive/02-04_genetic-algorithms

Solution presentation for FJSP

Chromosome: operator sequence & machine allocation components

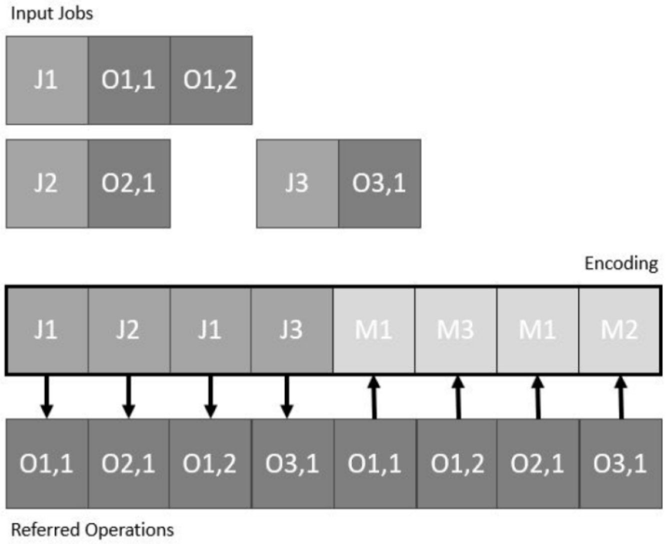


Figure 4.2: Example encoding format of a job schedule.

- The operation sequence determines the order in which operations should be scheduled
- The machine allocation determines on which an operation is scheduled.
  - O_{11} is scheduled on M1
  - O_{12} on M3 …

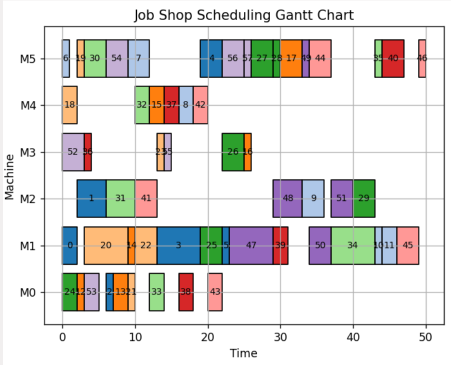
→ This enables a representation that is always feasible for FJSP

TU/e

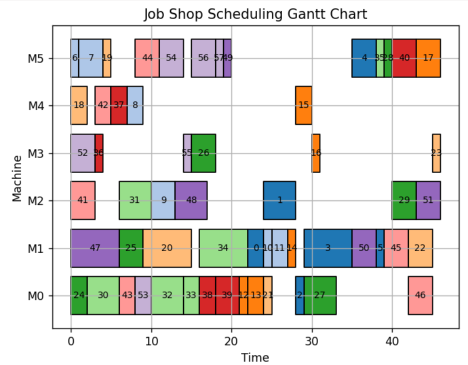
Genetic Algorithm

Crossover

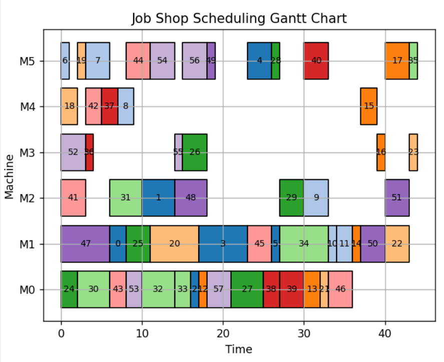
- Machine selection crossover → maintain certain machine allocations and fill remaining with other solution
- Operation sequence crossover → preserve relative scheduling random selected jobs and fill with other solution



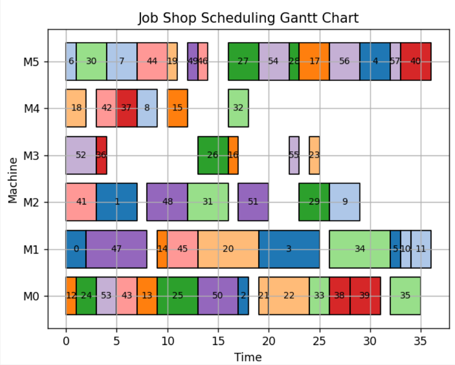
Gen 0 – makespan: 50



Gen 2 – makespan 46

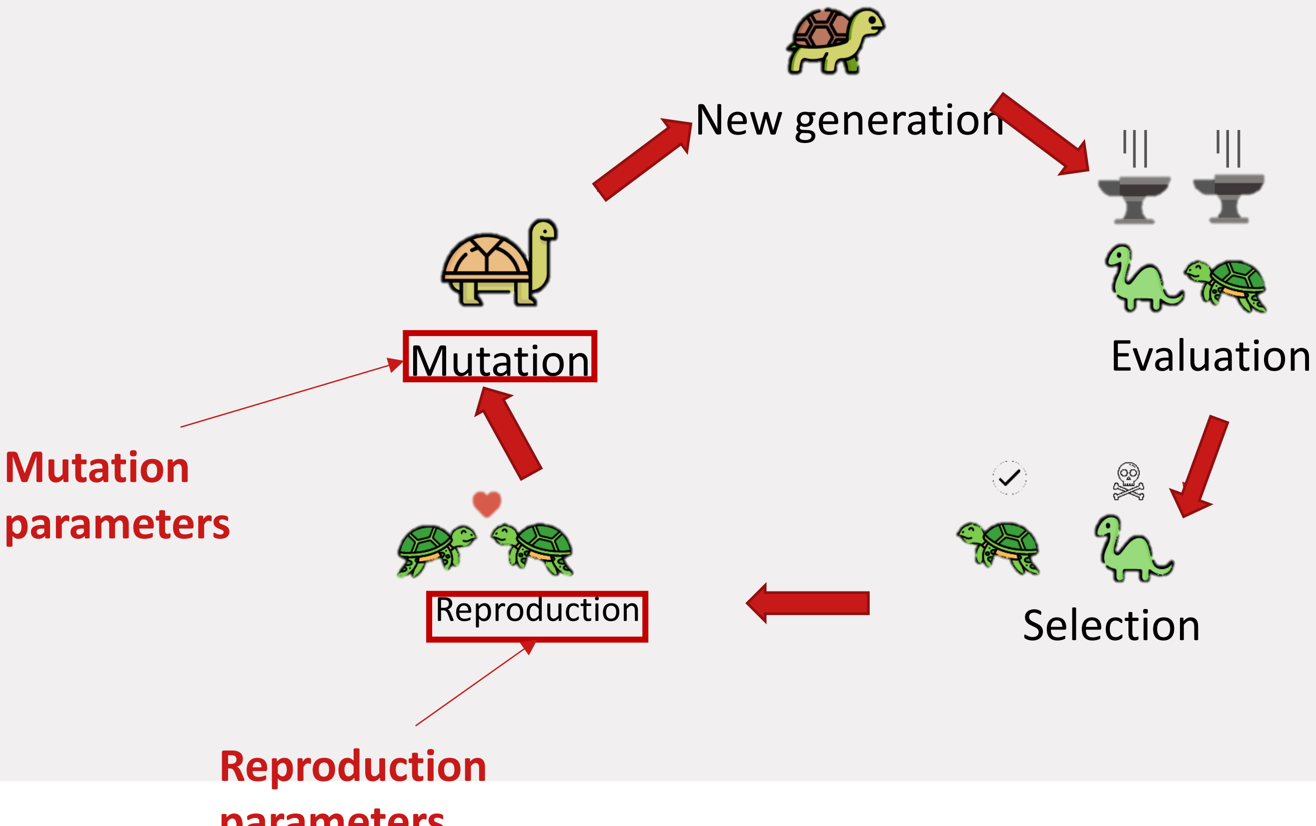


Gen 8 – makespan 44

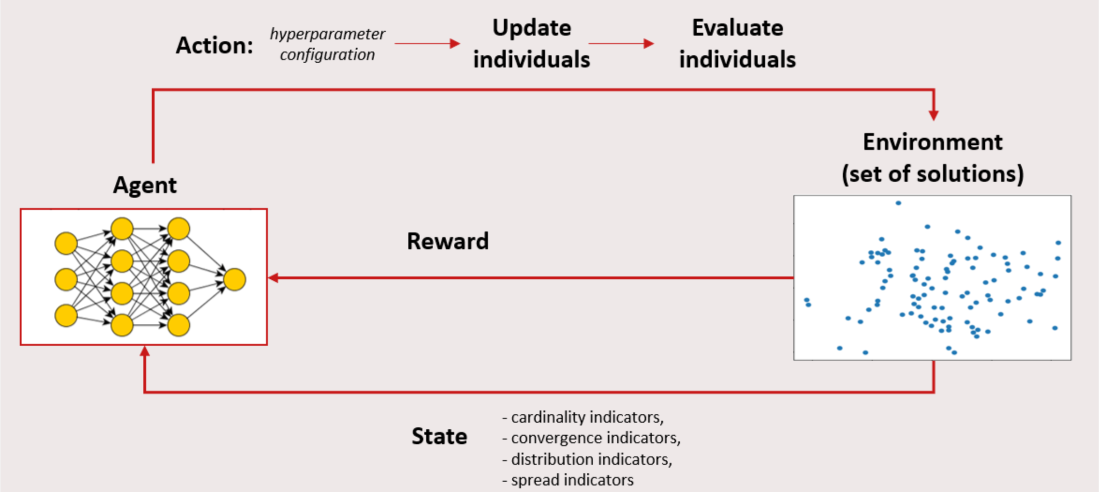


Gen 50 – makespan 36

SLEGA: Online control of Genetic Algorithm with DRL



SLEGA: GA controlled with DRL



MDP formulation

State:

quality of individuals (mean, max, standard deviation)

number of generations left, stagnation count

Actions:

individual mutation rate, crossover rate and gene mutation rate

Reward:

Objective value increased in new generation

Experiments

Experiments

Experiment	FJSP type	Datasets	Objective
1	Vanilla FJSP	Literature & Custom	Makespan
2	FJSP + SDST	Literature & Custom	Makespan
3	FJSP-WF (SDST, release date, night time)	Custom	Makespan

Criteria	SLEGA	E2E-DRL
Performance	?	?
Runtime	?	?
Scalable	?	?
Robust	?	?

Dataset

Benchmark datasets

Real-world data (WF)

instance	n	m	h_i	$\lvert M_{i,k}\rvert$	$p_{i,k,j}$	LB	UB	CP
mk01	10	6	[5, 7]	3	[1,7]	36	39	40
mk02	10	6	[5, 7]	6	[1,7]	24	26	27
mk03	15	8	10	5	[1,20]	204	204	204
mk04	15	8	[3,10]	3	[1,10]	48	60	60
mk05	15	4	[5,10]	2	[5,10]	168	172	174
mk06	10	15	15	5	[1,10]	33	58	59
mk07	20	5	5	5	[1,20]	133	139	143
mk08	20	10	[5,15]	2	[5,20]	523	523	523
mk09	20	10	[10,15]	5	[5,20]	299	307	307
mk10	20	15	[10,15]	5	[5,20]	165	197	214

Table 5.3: Brandimarte FJSP instances description

Instance	n	m		s,i,j,k,l	Flexibility	LB	UB	CP
1	2	2	2	[3,8]	Total	66	66	66
2	2	2	2	[3,10]	Partial	107	107	107
3	3	2	2	[6,15]	Partial	212	221	221
4	3	2	2	[8,21]	Partial	331	355	355
5	3	2	2	[3,6]	Total	107	119	119
6	3	2	2	[5,18]	Partial	310	320	320
7	3	5	3	[8,23]	Total	397	397	397
8	3	4	3	[4,13]	Total	216	253	253
9	3	3	3	[4,11]	Total	210	210	210
10	4	5	3	[10,28]	Partial	427	516	516
11	5	6	3	[8,24]	Partial	403	468	468
12	5	7	3	[9,26]	Partial	396	446	446
13	6	7	3	[9,30]	Partial	396	466	466
14	7	7	3	[10,31]	Partial	496	554	554
15	7	7	3	[10,30]	Partial	414	514	541
16	8	7	3	[10,30]	Partial	614	635	634
17	8	7	4	[10,31]	Partial	764	879	931
18	9	8	4	[10,30]	Partial	764	884	884
19	11	8	4	[10,30]	Partial	807	1088	1070
20	12	8	4	[10,33]	Partial	944	1267	1208

Table 5.6: Fattahi dataset description

E2E-DRL and SLEGA: Training

Train both our SO-SLEGA and E2E-DRL approaches on 100 different FJSP instances.

We do so for 5 different FJSP sizes from sodata, 10x05, 15x10, 20x05, 20x10, and a mix of the instance sizes.

Results - Experiment 1 – Vanilla FJSP

Heuristics rules

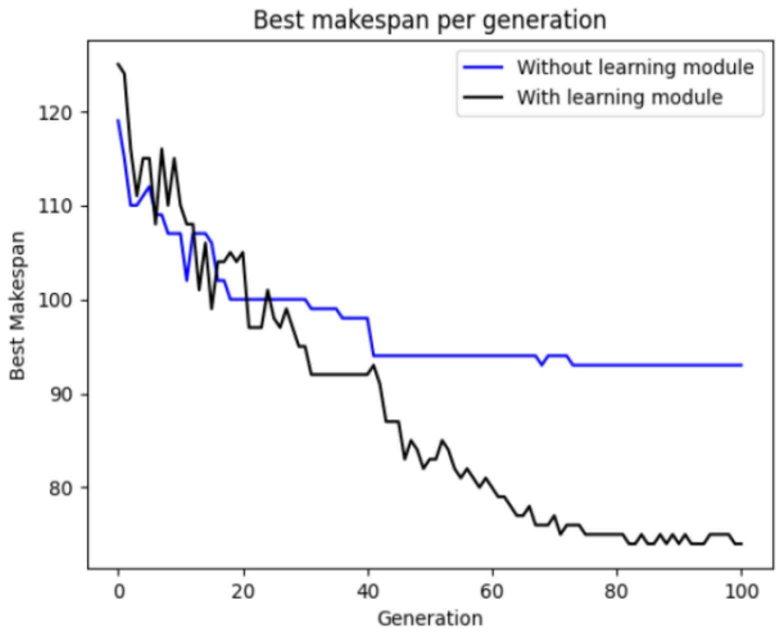
Vanilla GA

Model trained on different sized instances

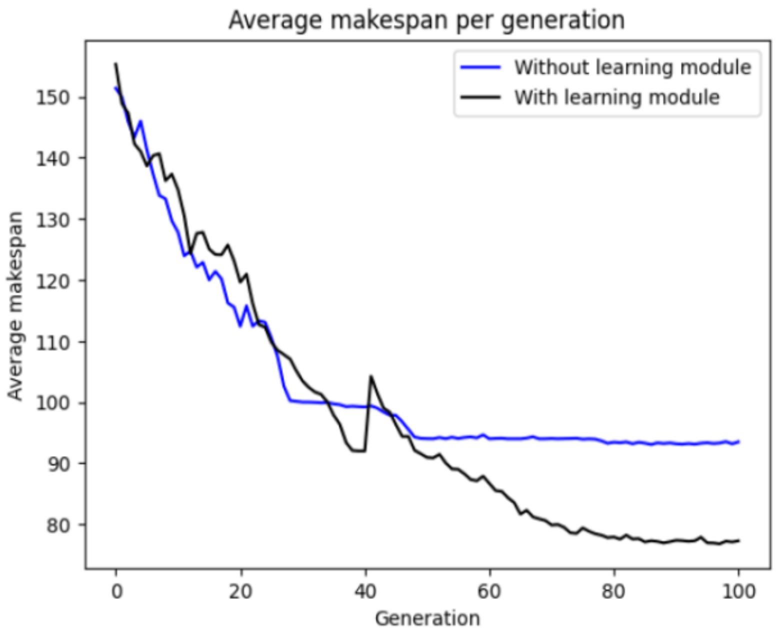
Approach		mkdata			edata		rdata		vdata	
		$\hat{C}_{\max}$	*Gopt*	$\hat{t}(s)$	$\hat{C}_{\max}$	*Gopt*	$\hat{C}_{\max}$	*Gopt*	$\hat{C}_{\max}$	*Gopt*
OPT		163.3	-	-	1005	-	923	-	807.9	-
SPT		283.2	73.48%	-	1305.35	29.89%	1184.80	28.36%	-	-
MOR		202.3	23.89%	-	1211.2	20.52%	1064.0	15.28%	-	-
MWKR		200.17	22.58%	-	1179.9	17.40%	1046.2	13.35%	-	-
FIFO		206.1	26.20%	-	1255.46	24.92%	1082.1	17.24%	-	-
RANDOM		637.5	290.39%	60.00	1225.9	21.98%	1212.9	31.41%	1041.7	28.94%
GREEDY		484.9	196.94%	1.30	1290.5	28.41%	1150.4	24.64%	894.5	10.72%
SO-GA		192.2	17.09%	30.02	1144.6	13.89%	1105.3	19.75%	953.4	18.01%
SLGA (R. Chen et al., 2020)		181.3	11.02%	-	-	-	-	-	-	-
DRL-G	10x05	200.1	22.54%	1.22	1193.1	18.72%	1049.7	13.73%	856.1	5.97%
	15x10	200.3	22.66%	1.23	1197.5	19.15%	1054.3	14.23%	858.0	6.20%
	20x05	220.1	34.78%	1.22	1269.5	26.32%	1125.1	21.90%	897.4	11.08%
	20x10	199.3	22.05%	1.21	1192.5	18.66%	1046.2	13.35%	841.3	4.13%
	Mixed	198.0	21.25%	1.29	1218.0	21.19%	1056.3	14.44%	845.0	4.59%
DRL-S	10x05	194.6	19.16%	4.60	1139.5	13.38%	1009.0	9.32%	827.6	2.44%
	15x10	193.1	18.25%	4.32	1144.9	13.92%	1008.2	9.23%	828.8	2.59%
	20x05	208.1	27.37%	4.25	1176.0	17.01%	1049.1	13.66%	857.8	6.18%
	20x10	195.2	19.53%	4.24	1152.85	14.71%	1015.5	10.02%	830.9	2.85%
	Mixed	196.8	20.51%	5.11	1107.5	10.20%	990.0	7.26%	831.1	2.87%
SO-SLEGA	10x05	180.7	10.66%	29.83	1103.5	9.80%	1047.8	13.52%	894.5	10.72%
	15x10	184.5	12.98%	29.62	1110.3	10.48%	1061.0	14.95%	914.7	13.22%
	20x05	180.1	10.29%	34.91	1097.7	9.22%	1040.3	12.71%	895.4	10.83%
	20x10	180.5	10.53%	33.23	1099.5	9.40%	1034.9	12.12%	888.13	9.93%
	Mixed	179.6	10.23%	28.18	1111.8	10.63%	1069.3	15.85%	955.8	18.31%

Table 6.1: Results of different algorithms on literature test data.

GA vs SLEGA



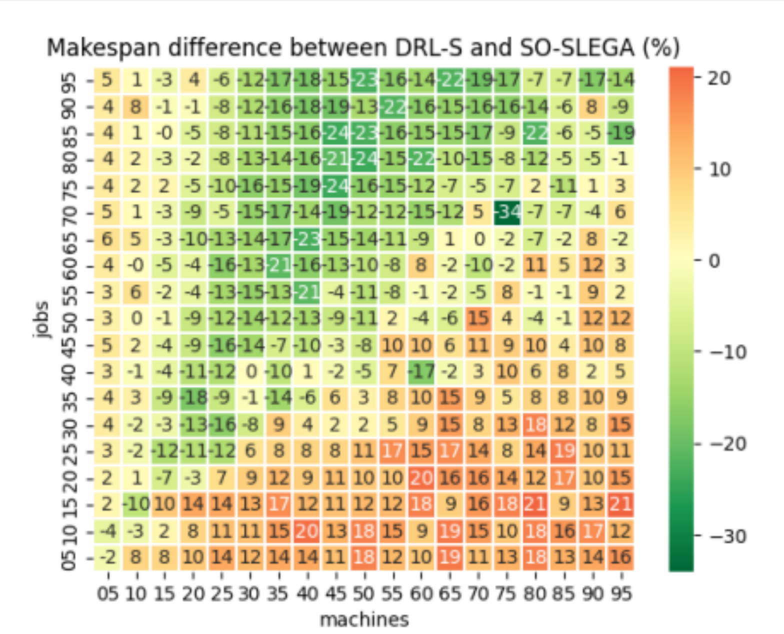
(a) Best makespan



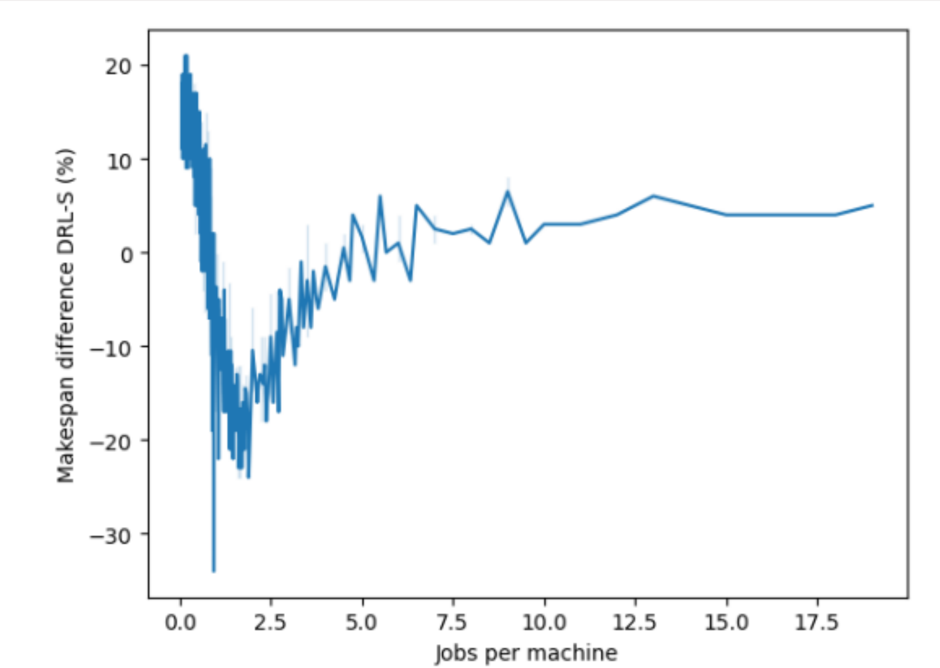
(b) Average makespan

Results - Experiment 1 – Vanilla FJSP

E2E-DRL and SO-SLEGA (both trained on 15x10 instances) on the cudata.



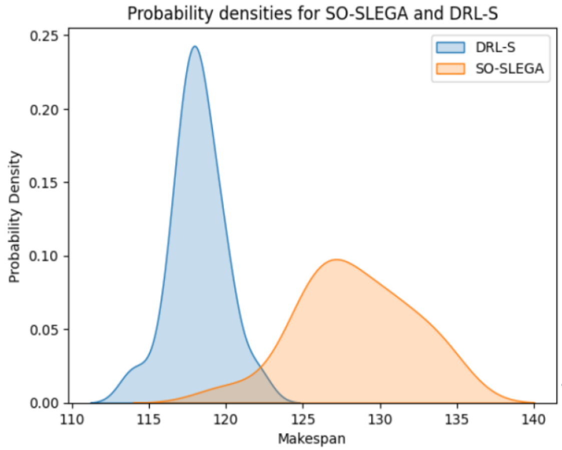
(a) Makespan



Results - Experiment 1 – Vanilla FJSP

Instance	J-M Ratio	OR-Tools LB	OR-Tools Computation Time	*Gopt*			
				OR-Tools	DRL-S 15x10	SO-SLEGA 15x10	SO-GA
15x80	0.2	99[1]	30s	0% (99[1])	21% (120)	0% (99[1])	0% (99[1])
80x50	1.6	115	6.5h	6% (122)	33% (153)	63% (188)	75% (203)
90x10	9.0	146	7.5h	314% (604)	371% (687)	336% (637)	476% (695)

[1]Optimal solution found. [2] DRL-S computes a solution within 1-60 seconds, whereas the (SLE)GAs take from 1-5 minutes.



Distribution of makespan for 50-job, 50-machine instances for DRL-S(15x10) and SO-SLEGA(15x10).

TU/e

Results - Experiment 2 - FJSP + SDSTs

- Solution quality
- Transferability

		FTdata				
		$\hat{C}_{\max}$	$Gopt$	$\hat{t}(s)$	Win Count	Average Rank
LB[2]		536.4	-	-	-	-
MWKR		787.4	46.81%	0.73	0	8.85
RANDOM		638.6	19.06%	60.00	10	3.75
GREEDY		667.5	24.45%	0.92	0	7.30
SO-GA		607.6	13.28%	59.24	4	5.10
DRL-G	10x05	643.2	19.91%	0.08	3	5.25
	15x10	634.9	18.37%	0.08		
	20x05	681.3	27.03%	0.08		
	20x10	648.4	20.89%	0.08		
DRL-S	10x05	577.0	7.57%	0.23	7	2.70
	15x10	572.8	6.80%	0.22		
	20x05	594.6	10.86%	0.21		
	20x10	576.3	7.45%	0.24		
DRL-S[1]	10x05	580.0	8.04%	0.22	7	3.10
	15x10	580.3	8.18%	0.22		
	20x05	589.9	9.97%	0.25		
	20x10	583.3	8.74%	0.22		
SO-SLEGA	10x05	543.3	1.29%	56.60	15	1.25
	15x10	568.4	5.98%	42.22		
	20x05	556.4	3.74%	66.08		
	20x10	549.4	2.42%	70.22		
SO-SLEGA[1]	10x05	547.7	2.12%	68.83	16	1.20
	15x10	547.1	1.99%	55.21		
	20x05	552.2	2.96%	66.92		
	20x10	545.5	1.71%	68.20		

[1]Models trained on vanilla FJSP (Experiment 1), [2]lower bound calculated using best makespan found over variou algorithms.

Table 6.3: Results of different algorithms on SDST literature test data.

Results - Experiment 3 - WFdata

We train the E2E-DRL and SO-SLEGA approaches on instance sets of different sizes.

- Wfdata: Release dates, deadlines, night times

Instances	Training				Validation					
	E2E-DRL		SO-SLEGA		Inference Time			Average Makespan		
	Duration	Iteration	Duration	Timestep	DRL-G	DRL-S (20x)	SO-SLEGA	DRL-G	DRL-S (20x)	SO-SLEGA
17x02	0.8h	840	3.3h	14000	0.2s	2.8s	48.9s	142787	127219	101122
42x02	2.4h	460	6.2h	24000	0.5s	3.3s	93.8s	281604	282146	223935
64x04	4.8h	160	7.5h	16000	0.8s	3.8s	111.3s	243945	230984	192755
88x08	7.4h	460	10.9h	11000	1.1s	4.3s	142.1s	202454	185219	165779

Results - Experiment 3 - WFdata

WFdata: with additional constraints: release dates, deadlines, night times

Instances where the number of jobs ranges between 5 and 100 and the number of machines ranges between 2 and 10 are considered.

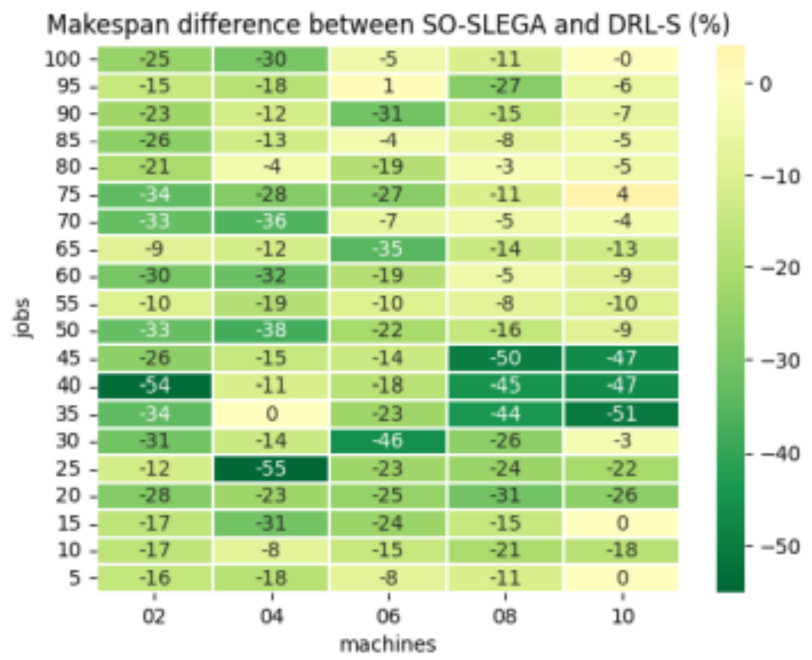
An instance of size 100x10 matches the industry scale.

		WFdata				
		$\hat{C}_{\max}$	$Gopt$	$\hat{t}(s)$	Win Count	Average Rank
LB[2]		132173	-	-	-	-
MWKR		266848	101.89%	6.22	1	6.85
RANDOM		195869	46.95%	10.00	3	5.29
GREEDY		164040	23.07%	15.63	24	3.16
SO-GA		161651	22.31%	37.45	13	3.14
DRL-G	17x02	177421	34.23%	1.68	1	3.96
	42x02	181551	37.44%	1.47		
	64x04	192354	45.53%	2.32		
	88x08	187690	42.00%	2.20		
DRL-S	17x02	175477	32.76%	5.46	5	3.51
	42x02	182804	38.31%	4.35		
	64x04	177147	34.03%	2.87		
	88x08	183488	38.82%	3.35		
	15x10[1]	188850	42.88%	2.88		
SO-SLEGA	17x02	138733	4.96%	35.92	81	1.21
	42x02	160726	21.60%	33.96		
	64x04	142333	7.69%	29.65		
	88x08	144898	9.63%	30.45		
	15x10[1]	143992	8.03%	39.54		

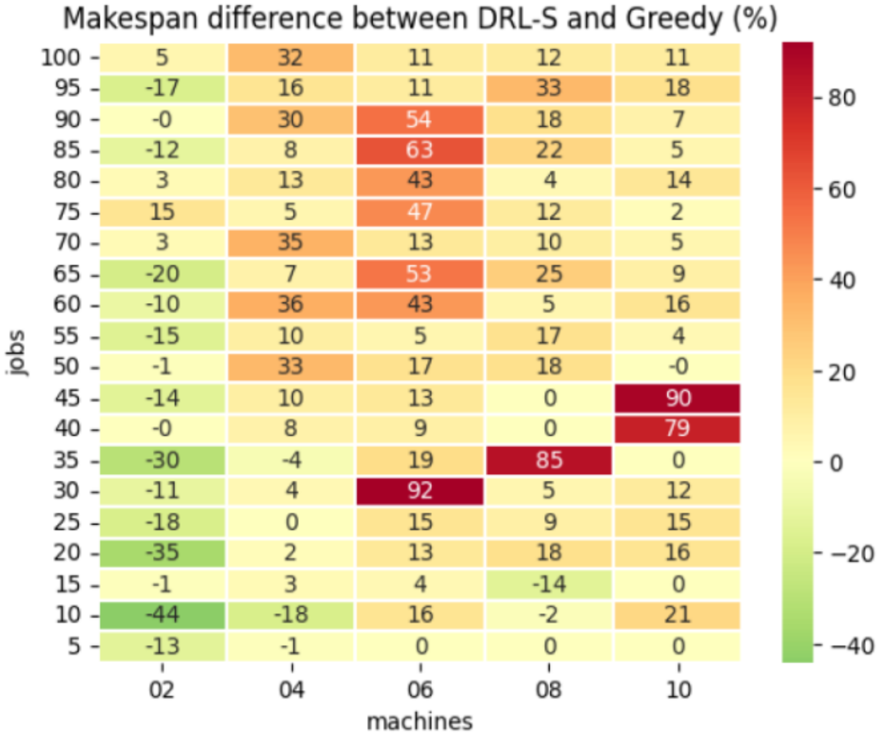
[1]Models trained on vanilla FJSP (Experiment 1), [2]lower bound calculated using best makespan found per instance a all algorithms.

Table 6.7: Results of different algorithms on WF-specific instances.

Results - Experiment 3 - WFdata



(a) Makespan



(b) DRL-S (17x02)

When E2E-DRL falls short?

Index	Instance Type	Average makespan GREEDY	SO-SLEGA	DRL-S	DRL-S stack. feat.	DRL-S all feat.	Δ
1	FJSP	48170	45051	45821	45821	46556	770 (1.7%)
2	(1) with SDSTs	102778	93992	119522	118148	128335	25530 (20.5%)
3	(2) with Release Dates	115135	103773	129749	126371	132535	22598 (17.9%)
4	(3) with Night Times	164040	138733	184627	175447	175447	36714 (20.9%)
5	(1) with Night Times	71760	64261	68276	67727	73552	3466 (5.4%)

Table 6.9: Ablation study for FJSP characteristics and E2E-DRL features.

Hybrid approach vs E2E DRL

Criteria	SLEGA	E2E-DRL
Performance	Performs well generally speaking.	Fails to perform when under more complicated constraints and instances.
Runtime	Fast enough.	Almost instant.
Scalability	Generalizable to larger instances. Can be parallelized in order to support scalability.	Training on large instances does not work. Retraining is necessary for different instance characteristics.
Transferability/ robustness	Can handle various instance types.	Fails to deal with complicated instance types.

Are DRLs there?

- End-to-end DRL
  - Fast (on solving)
  - Hard to scale, hard to handle heavily constrained problems
- Hybrid approaches work
  - learning to guide search/speed up solution finding
  - highly transferable/generalizable

Coming soon: Job Shop Scheduling Benchmark

A open sourced repo for benchmarking scheduling solutions

- Benchmark instances of variants of JSP problems + various solution methods
- Environment of developing other (learning based) solution methods

Release date V1: August 15, 2023:
https://github.com/RobbertReijnen/Job_Shop_Scheduling_Benchmark

Solutions methods	Job Shop	Flow Shop	Flexible	SDST	Assembly operations	Online Arrivals
Heuristics	✓	✓	✓	✓	✓	
Dispatching Rules	✓	✓	✓	✓	✓	✓
GA: Genetic Algorithm	✓	✓	✓	✓	✓	
SLEGA: GA with DRL	✓	✓	✓	✓	✓	
DRL – learning to dispatch	✓	✓	✓	✓	✓	✓
E2E DRL with GNN	✓	✓	✓	✓		✓

The need of benchmarks

- The first AI for TSP competition @ IJCAI 2021
  https://tspcompetition.com/

  Zhang et al., The first AI4TSP competition: Learning to solve stochastic routing problems, Artificial Intelligence. 2023
- EURO Meets NeurIPS 2022 Vehicle Routing Competition,

  https://euro-neurips-vrp-2022.challenges.ortec.com/

  Kool et al., EURO Meets NeurIPS 2022 Vehicle Routing Competition. 2022

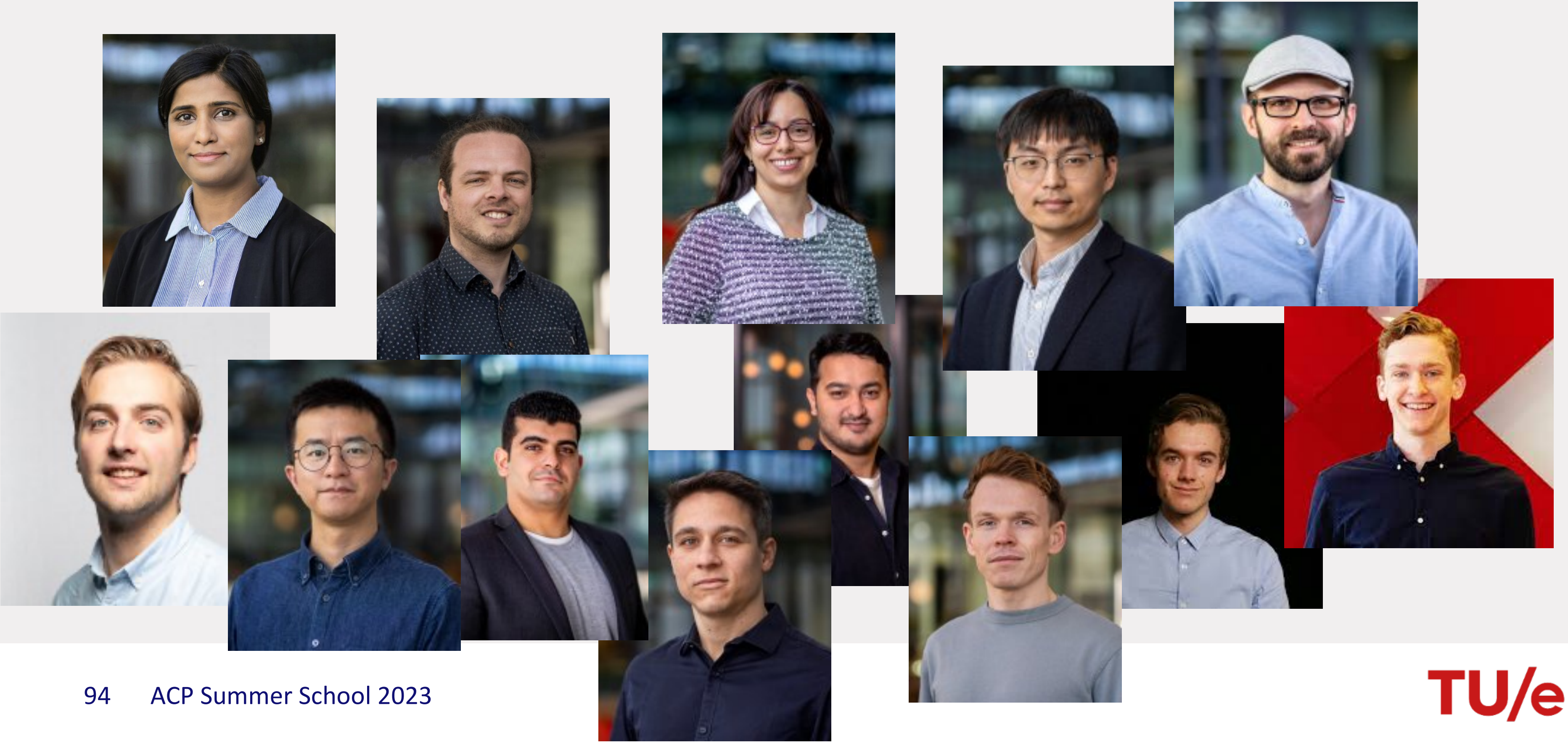
References

- Smit et al., Learning to Be Efficient and Fair for Collaborative Order Picking, working paper (happy to share)
- Begnardi et al., Deep Reinforcement Learning for Two-sided Online Bipartite Matching in Collaborative Order Picking, under review (happy to share)
- Van Straaten et al., Flexible Job-Shop Scheduling under heavy constraints with learning based approaches, working paper (happy to share)
- Zhang et al., The first AI4TSP competition: Learning to solve stochastic routing problems, Artificial Intelligence. 2023
- Kool et al., EURO Meets NeurIPS 2022 Vehicle Routing Competition. 2022
- Ya et al., Revisit the Algorithm Selection Problem for TSP with Spatial Information Enhanced Graph Neural Networks. 2023
- Da Costa et al., Policies for the dynamic traveling maintainer problem with alerts, European Journal of Operational Research, 2023
- Beeks et al., Deep Reinforcement Learning for a Multi-Objective Online Order Batching Problem, ICAPS 2022
- Cals et al., Solving the online batching problem using deep reinforcement learning, CAIE 2021

TU/e

- Afshar et al., An Automated Deep Reinforcement Learning Pipeline for Dynamic Pricing, IEEE Transaction on AI, 2022
- Peer et al., Shunting Trains with Deep Reinforcement Learning, IEEE SMC, 2018
- Wu et al., Neural Airport Ground Handling, IEEE Transactions on Intelligent Transportation Systems, 2023
- Sutton and Barto, Reinforcement learning: an introduction
- Schulman et al. Proximal policy optimization algorithms, 2017
- Alomrani et al., Deep Policies for Online Bipartite Matching: A Reinforcement Learning Approach, TMLR, 2022
- Xu et al. Prediction-guided multi-objective reinforcement learning for continuous robot control. ICML 2020.
- Afshar et al., *Automated Reinforcement Learning: An Overview.* 2022
- Vos and Verwer, Optimal Decision Tree Policies for Markov Decision Processes, AAAI 2023
- Song et al., Flexible job shop scheduling via graph neural network and deep reinforcement learning. IEEE Transactions on Industrial Informatics, 2022
- Chen et al., A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. CAIE. 2020

Team AI for decision-making @TU/e



TU/e