

# Generalizing Beyond Suboptimality: Offline Reinforcement Learning Learns Effective Scheduling through Random Solutions

Anonymous authors

Paper under double-blind review

## Abstract

Online reinforcement learning (RL) approaches have demonstrated strong performance on Job Shop Scheduling (JSP) and Flexible JSP (FJSP) problems by learning scheduling policies through direct interaction with simulated environments. However, these methods often require extensive training interactions, limiting their sample efficiency and practical applicability.

Motivated by this challenge, we introduce **Conservative Discrete Quantile Actor-Critic (CDQAC)**, an offline RL algorithm that learns effective scheduling policies directly from static, suboptimal datasets. CDQAC couples a quantile-based critic with delayed policy updates to estimate the return distribution of machine–operation pairs. Extensive experiments on JSP and FJSP benchmarks demonstrate that CDQAC consistently outperforms the data-generating heuristics, surpasses state-of-the-art offline and online RL baselines, and is highly sample efficient, requiring only 1 to 5% of the original dataset to learn high-quality policies. Our analysis suggests that, in scheduling, offline RL performance is governed mainly by state-action coverage rather than the quality of individual trajectories. Scheduling couples a dense reward aligned with the makespan objective with equal-length trajectories across heuristics, enabling effective learning from a broad range of behaviors. Consistent with this observation, datasets generated by a simple random heuristic with broader coverage let it outperform policies trained on datasets produced by stronger heuristics such as Genetic Algorithms.

## 1 Introduction

The Job Shop Scheduling Problem (JSP) and Flexible Job Shop Scheduling Problem (FJSP) are fundamental challenges in manufacturing and industrial operations (Bhatt & Chauhan, 2015), where the goal is to optimally schedule *jobs* on available *machines* to minimize objectives such as total completion time (makespan). Exact methods such as Constraint Programming (CP) (Da Col & Teppan, 2022) and Mathematical Programming (Fan & Su, 2022) guarantee optimality but face scalability issues for large-sized instances. Therefore, in practice, heuristic methods such as Genetic Algorithms (GA) (Bhatt & Chauhan, 2015) and Priority Dispatching Rules (PDRs) (Veronique Sels & Vanhoucke, 2012) are often preferred, as they can find acceptable solutions in reasonable time.

Deep reinforcement learning (RL) has emerged as a promising approach for learning dispatching rules; methods like Learning-to-Dispatch (L2D) Zhang et al. (2020) can outperform standard dispatching rules, infer high quality solutions orders of magnitude faster than exact solvers, and generalize to larger instances. However, because they train from scratch via trial-and-error, these online methods suffer from sample inefficiency, often requiring a vast amount of simulator interactions to converge. Conversely, heuristics such as PDR and Genetic Algorithms (GA) are widely used and can easily generate substantial, albeit inherently suboptimal, training data. Offline RL offers a paradigm to leverage these static datasets by estimating value rather than merely imitating actions, enabling it to learn a new policy not present in the training dataset. Remmerden et al.

(2025) recently introduced Offline-LD as the first offline RL method for JSP, where it outperformed L2D using only 100 instances. Crucially, the reliance of Offline-LD on computationally expensive CP solutions limits its scalability, leaving the potential of learning from vast amounts of suboptimal heuristic data underutilized.

We argue that scheduling problems such as JSP and FJSP are particularly well suited to learning from suboptimal—even random—data. Unlike prior offline RL applications such as robotics (Levine et al., 2020; Figueiredo Prudencio et al., 2024) where the reward is sparse or an approximation of the true objective, scheduling provides a dense reward signal that is aligned exactly with the optimization objective through partial makespan increases (Smit et al., 2025), allowing every transition to carry a calibrated value signal. Furthermore, constructive scheduling policies, whether expert, heuristic, or random, explore from the same states and always generate trajectories of identical length, equal to the number of operations to be scheduled. As a result, scheduling avoids the trajectory-length bias that Li et al. (2023) shows can make offline RL favor expert data. Together, these properties suggest that offline RL performance on scheduling is governed by *state-action coverage* (Schweighofer et al., 2022) rather than *data quality*.

With this motivation, we propose **Conservative Discrete Quantile Actor-Critic** (CDQAC), a novel offline RL method for scheduling, which learns effective scheduling policies from **suboptimal examples** generated by a wide range of heuristics. CDQAC learns an approximated representation of the value of each action from which it can generalize a new policy that can outperform the heuristic that generated the data. CDQAC achieves this through a quantile-based critic, with a new dueling architecture. This critic provides value estimates that guide the actor, while a delayed policy update prevents the propagation of early noisy critic predictions, ensuring stable joint learning of the policy and value function.

Our contributions are as follows: (1) We introduce CDQAC, a offline RL method that demonstrates very strong performance on both JSP and FJSP, by solely training on suboptimal examples. CDQAC outperforms the tested online RL methods while using only suboptimally generated random data, removing the need for both a training simulator and expert demonstrations. (2) CDQAC integrates a quantile critic with a dueling architecture that uses *separated* inputs for the value and advantage streams — tailored to scheduling’s variable-size action spaces — and a delayed policy update that prevents bootstrapped error propagation in offline discrete RL. (3) We show that random data is the *strongest* training source for offline RL on scheduling problems. This follows from two structural properties of scheduling: a dense, objective-aligned reward and equal-length trajectories across all heuristics. Under these conditions, state-action coverage rather than data quality becomes the binding constraint on offline performance, and a random policy maximizes coverage by construction. (4) CDQAC is highly sample-efficient, achieving competitive performance with 10–25 training instances, which is typically 1–5% of the data used by online RL baselines.

## 2 Preliminaries

**JSP & FJSP.** We formulate the Job Shop Scheduling (JSP) and Flexible Job Shop Scheduling Problem (FJSP) as follows. Given a set of  $n$  jobs, represented as  $\mathcal{J}$ , and a set of  $m$  machines, represented as  $\mathcal{M}$ , each job  $J_i \in \mathcal{J}$  has  $n_i$  operations. These operations  $\mathcal{O}_i = \{O_{i,1}, O_{i,2}, \dots, O_{i,n_i}\}$  must be processed in order, forming a precedence constraint. In JSP, each operation  $O_{i,j}$  can only be processed by a single machine, whereas in FJSP,  $O_{i,j}$  can be processed on any machine in its set of compatible available machines  $\mathcal{M}_{i,j} \subseteq \mathcal{M}$ . Each machine  $M_k \in \mathcal{M}_{i,j}$  has a specific processing time for an operation  $O_{i,j}$  denoted as  $p_{i,j}^k$ , where  $p_{i,j}^k > 0$ . The objective is to minimize the makespan, defined as the completion of the last operation  $C_{\max} = \max_{O_{i,j} \in \mathcal{O}} C(O_{i,j})$ , where  $C(O_{i,j})$  represents the completion time of operation  $O_{i,j}$ .

**Offline Reinforcement Learning.** We formalize FJSP and JSP as a Markov Decision Process (MDP) denoted as  $\mathbf{M}_{\text{MDP}} = \langle \mathcal{S}, \mathcal{A}(s_t), P, R, \gamma \rangle$ . A state  $s_t \in \mathcal{S}$  represents the progress of the current schedule in the timestep  $t$ , and includes all operations  $O_{i,j} \in \mathcal{O}_t$  that are available to be scheduled on machines  $M_k \in \mathcal{M}_t$ , whereby  $\mathcal{M}_t$  only contains machines that are free at timestep  $t$ . The action space  $a_t \in \mathcal{A}(s_t)$  corresponds to all available machine-operation pairs  $(O_{i,j}, M_k)$  at  $t$ .  $P$  is the transition function and determines the next state  $s_{t+1}$  on the selected machine-operation pair  $(O_{i,j}, M_k)$ , whereby unavailable pairs, due to  $M_k$  being selected, being removed and new available pairs added. The reward  $r_t$  is the negative increase in the (partial) makespan resulting from action  $a_t$ :  $r_t = \max_{O_{i,j} \in \mathcal{O}} C(O_{i,j}, s_t) - \max_{O_{i,j} \in \mathcal{O}} C(O_{i,j}, s_{t+1})$ ,  $\gamma$  is the discount

factor that determines the importance of future rewards. We set  $\gamma = 1$ . In offline RL, a policy  $\pi(a|s)$  is learned through a static dataset  $D = \{(s, a, r(s, a), s')_i\}$ , where  $s'$  is the next state.  $D$  is generated through one or more behavioral policies  $\pi_\beta$ .

### 3 Related Work

**Learning-based methods for Scheduling Problems.** Deep reinforcement learning (DRL) has emerged as a competitive alternative to traditional dispatching rules and exact solvers for JSP and FJSP, offering near-real-time inference, strong cross-instance generalization, and consistently outperforming standard priority dispatching rules (Smit et al., 2025). The dominant paradigm in this line of work is online DRL: a constructive policy is trained from scratch through repeated interaction with a simulated scheduling environment. L2D (Zhang et al., 2020) established this paradigm for JSP, formulating scheduling on a disjunctive graph and training a GNN policy with PPO to surpass classical heuristics, with subsequent extensions to multi-agent settings (Park et al., 2021). The constructive online setup was later adapted to FJSP, most notably by FJSP-DRL (Song et al., 2023), which uses a heterogeneous graph encoder, and DANIEL (Wang et al., 2023), which jointly reasons over operations and machines through a dual attention network. Beyond purely constructive policies, complementary directions augment the online loop with curriculum strategies (Iklavov et al., 2023), behavior cloning losses against CP demonstrations (Tassel et al., 2023), RL-guided improvement heuristics (Zhang et al., 2024a;b), and self-supervised constructive learning (Corsini et al., 2024; Pirnay & Grimm, 2024). All of these methods have shown their competitive performance as the traditional heuristics on JSP and FJSP benchmarks. However, they still require a simulated environment during training and therefore inherit the sample inefficiency that characterizes online RL. Building on the demonstrated effectiveness of online DRL for JSP and FJSP, this paper investigates whether an offline approach can attain comparable performance while training purely on static, suboptimal datasets — removing the need for any environment interaction. The only prior offline RL method in this space, Offline-LD (Remmerden et al., 2025), partially addresses this question but is restricted to JSP and requires (near-)optimal CP solutions as training data, a limitation we revisit in the next subsection.

**Offline Reinforcement Learning.** Most offline RL methods target continuous action spaces (An et al., 2021; Kostrikov et al., 2021; Jackson et al., 2026), while discrete-action methods are typically benchmarked on settings with fixed state and action sizes (Kumar et al., 2023). Both assumptions are incompatible with FJSP and JSP, whose action spaces are discrete, variable-size, and instance-dependent. This rules out continuous-action methods such as TD3+BC and SPOT, as well as transformer-based sequence models (Chen et al., 2021; Janner et al., 2021), which require fixed state/action dimensions. Among methods compatible with our setting, Conservative Q-Learning (CQL) (Kumar et al., 2020) prevents overestimation of out-of-distribution actions through value regularization and has been applied to large discrete-action problems (Kumar et al., 2023). Distributional offline RL was explored by CODAC (Ma et al., 2021), which combines quantile-based distributional RL with conservative penalties for risk-sensitive continuous control. CDQAC differs from CODAC in three respects: (1) we target discrete, variable-size scheduling actions rather than continuous control; (2) we apply CQL conservatism on the scalar mean of the quantile distribution rather than penalizing the full return distribution, since our objective is risk-neutral makespan minimization; and (3) we introduce a separated-input dueling architecture and delayed policy updates tailored to scheduling. A consistent finding across these benchmarks is a data-quality ordering: noisy-expert datasets dominate, while uniformly random data is the weakest training source (Schweighofer et al., 2022; Kumar et al., 2022). This ordering reflects the structural properties of the domains in which offline RL is usually studied—robotics and Atari—where rewards are sparse or proxy and random policies produce short, failure-terminated trajectories. Scheduling is a different setting: it provides a dense, objective-aligned reward and equal-length trajectories regardless of the generating heuristic. Under these conditions, state-action coverage rather than data quality becomes the governing factor, which makes diverse and even random data a natural strength rather than a liability. We develop this argument in detail in Sect. 5.2 and confirm it empirically. Offline RL has only recently been applied to scheduling. Offline-LD (Remmerden et al., 2025) first demonstrated offline RL for JSP using (near-)optimal constraint programming solutions, but is limited to JSP and requires expert-quality data. CDQAC extends offline RL to both JSP and FJSP and, unlike prior work, learns effective policies from purely suboptimal data — including data generated by a random policy. This distinguishes our setting from

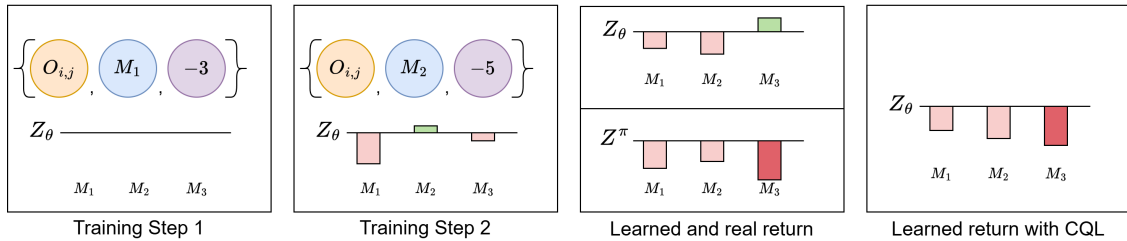


Figure 1: Illustrative example of overestimating OOD actions. In training steps 1 and 2 examples are shown of negative outcomes of pairing operation  $O_{i,j}$  with either machine  $M_1$ , with a reward of  $-3$ , or  $M_2$ , with a reward of  $-5$ , learning that  $M_3$  results in the best outcome, since the combination  $(O_{i,j}, M_3)$  does not exist in the dataset. The real return  $Z_\pi$  shows that  $M_3$  results in the worst outcome. CQL ensures OOD actions are not overestimated, in comparison to actions in the dataset.

imitation learning, or behavioral cloning, which imitates near-optimal demonstrations and cannot improve beyond the data generator (Luo et al., 2023; Drakulic et al., 2023).

#### 4 Conservative Discrete Quantile Actor-Critic for Scheduling

Our goal is to learn a scheduling policy  $\pi_\psi$  from a static dataset  $D$  that surpasses the behavioral policies  $\pi_\beta$  that generated it.  $\pi_\beta$  may be any (possibly non-Markovian) heuristic, such as PDRs, genetic algorithms, or random schedulers. To outperform  $\pi_\beta$ , the learner must estimate accurate state-action values  $Q_\theta(s, a)$  in  $D$  and “stitch” high-value segments into a better policy. This differs from Behavioral Cloning, which learns to imitate the actions of  $\pi_\beta$ . Because  $\pi_\psi$  is updated solely via  $Q_\theta$ , we employ a critic that must (1) model the *return distribution* for state-action pairs observed in  $D$  and (2) remain *conservative* on out-of-distribution (OOD) actions to avoid overestimation under distributional shift, while still enabling improvement beyond the data. For this purpose, we propose Conservative Discrete Quantile Actor-Critic (CDQAC), an offline RL approach for JSP and FJSP. CDQAC integrates a **quantile critic** with a delayed policy update, enabling the learning of a scheduling policy from a dataset  $D$  composed of suboptimal examples, while still discovering policies that outperform those contained in  $D$ .

**Quantile Critic.** A single scheduling action induces a return distribution spread across several distinct outcomes: assigning operation  $O_{i,j}$  to machine  $M_k$  determines which subsequent (operation, machine) pairs become feasible, so the same action can yield substantially different makespans depending on later decisions. A scalar  $Q$ -value collapses this spread into a single expectation that need not correspond to any likely outcome. We therefore approximate the random return  $Z^\pi(s, a) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$  directly, rather than its expectation  $Q^\pi(s, a) = \mathbb{E}[Z^\pi(s, a)]$ , which has been shown to yield more accurate value estimates than standard  $Q$ -learning (Bellemare et al., 2017; Dabney et al., 2018).

To mitigate value overestimation, we parameterize the critic with two heads  $Z_{\theta_i}$ ,  $i \in \{1, 2\}$ , and use  $Z_\theta = \min(Z_{\theta_1}, Z_{\theta_2})$  in target computation (Christodoulou, 2019; Zhou et al., 2024). Each head approximates the return as a uniform mixture of  $N$  Dirac deltas at fixed fractions  $\tau_n = (2n-1)/(2N)$ ,  $n \in [1, \dots, N]$  (Dabney et al., 2018):

$$Z_{\theta_i}(s, a) = \frac{1}{N} \sum_{j=1}^N \delta(\theta_i^j(s, a)), \quad (1)$$

where  $\theta_i^j(s, a)$  predicts the  $j$ -th quantile and  $\delta$  is the Dirac delta. The corresponding distributional Bellman target is

$$\mathcal{T}Z(s, a) = r(s, a) + \gamma Z_{\theta_i}(s', a'), \quad s' \sim D, a' \sim \pi_\psi(\cdot | s'), \quad (2)$$

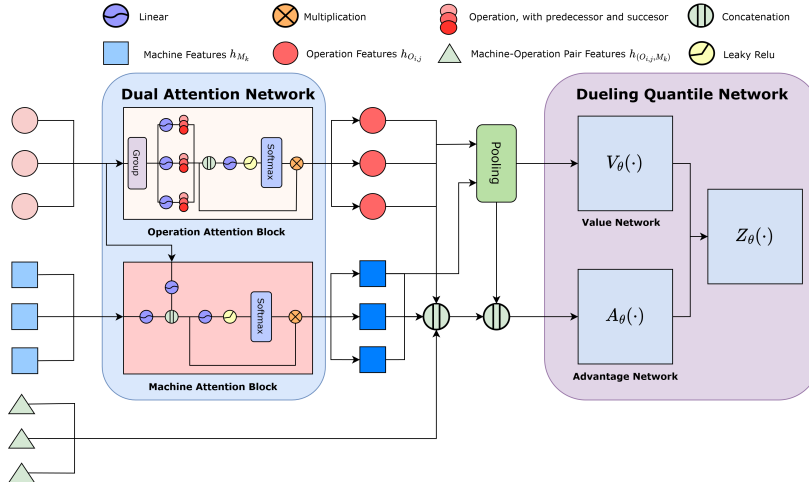


Figure 2: The network architecture. (Left) The Dual Attention Network (DAN) encodes the operations and machines. (Right) The Dueling Quantile Network uses these embeddings to learn the machine-operation pair, whereby it combines the Value  $V_\theta$  and Advantage  $A_\theta$  streams through Eq. 6.

where  $\hat{\theta}$  denotes the target network and  $a'$  is sampled from the current policy, tying the learned distribution to the policy being optimized. The temporal-difference (TD) loss is

$$\mathcal{L}_{\text{TD}}(\theta) = \mathbb{E}_{\substack{(s,a,r,s') \sim D \\ a' \sim \pi_\psi(\cdot | s')}} [\rho_\tau^H(\mathcal{T}Z(s, a) - Z_\theta(s, a))], \quad (3)$$

where  $\rho_\tau^H$  is the asymmetric quantile Huber loss (Dabney et al., 2018) applied jointly across all fractions  $\tau$ . The target network is updated by Polyak averaging,  $\hat{\theta} \leftarrow (1 - \rho)\hat{\theta} + \rho\theta$ . A scalar  $Q$ -value is recovered as  $Q_\theta^Z(s, a) = \mathbb{E}[Z_\theta(s, a)]$ .

**Conservative Q-Learning.** In the offline setting, CDQAC updates  $Z_\theta$  using targets that can involve actions without support in the static dataset  $D$ . This support mismatch, i.e. *distributional shift* between the state-action distribution in  $D$  and that induced by the learned policy, leads the critic to overestimate the values for out-of-distribution (OOD) actions, as illustrated in Fig. 1. This overestimation is not an issue for online RL, since it can explore these actions during training; however, offline RL cannot due to learning from a static dataset. To avoid this overestimation, we add Conservative Q-learning (CQL) (Kumar et al., 2020) to the loss of the critic. CQL penalizes overestimation of OOD actions, by introducing a regularization term used in combination with standard critic loss:

$$\mathcal{L}_Z(\theta) = \alpha_{\text{CQL}} \mathbb{E}_{s \sim D} \left[ \log \sum_{a' \in \mathcal{A}(s)} \exp(Q_\theta^Z(s, a')) - \mathbb{E}_{a \sim D} [Q_\theta^Z(s, a)] \right] + \mathcal{L}_{\text{TD}}(\theta), \quad (4)$$

where,  $\alpha_{\text{CQL}}$  determines the strength of the penalty, and  $\mathcal{L}_{\text{TD}}(\theta)$  is the loss in Eq. 3.

**Delayed Policy Updates.** CDQAC updates the critic using bootstrapped targets whose successor actions are sampled from the current policy  $\pi_\psi(\cdot | s')$ , while the policy is optimized against the critic's current estimates. Early in training, this coupling can amplify critic errors: noisy value estimates may increase the probability of poorly supported machine-operation pairs, which are then reused in subsequent Bellman targets. This issue is particularly harmful in offline scheduling, where the static dataset provides no online interaction to correct such errors.

We therefore update the policy only every  $\eta$  critic steps. This gives the quantile critic more updates to fit the return distribution and apply conservative regularization before it is used for policy improvement. Delayed policy updates are established in online continuous-control RL (Fujimoto & Gu, 2021); here, we use them as a stabilization mechanism for offline discrete actor-critic learning with variable-size combinatorial action

spaces. Our ablations show that this mechanism is important for preventing premature policy improvement from an unreliable critic. The policy loss is

$$\mathcal{L}_\pi(\psi) = \mathbb{E}_{\substack{s \sim D \\ a \sim \pi_\psi(\cdot | s)}} [-Q_\theta^Z(s, a) - \lambda \mathcal{H}[\pi_\psi(\cdot | s)]] , \quad (5)$$

where the entropy bonus encourages diversity over feasible machine–operation pairs and prevents premature policy collapse.

#### 4.1 Network Architecture

To encode an FJSP or JSP instance, we use a dual attention network (DAN), adapted from DANIEL (Wang et al., 2023), for both the policy network  $\pi_\psi$  and the quantile critic  $Z_\theta$ . Fig. 2 shows our network architecture. DAN processes two parallel attention streams over the relevant operations  $O_{i,j} \in \mathcal{O}_t$  and machines  $M_k \in \mathcal{M}_t$ , learning the complex relation between each machine–operation pair at timestep  $t$  and embedding them as  $h_{O_{i,j}}$  and  $h_{M_k}$ . A detailed explanation of DAN and the input features is provided in App. B.

From the machine embeddings  $h_{M_k}$  and operation embeddings  $h_{O_{i,j}}$ , we compute a global embedding  $h_G = \left[ \left( \frac{1}{|\mathcal{O}_t|} \sum_{O_{i,j} \in \mathcal{O}_t} h_{O_{i,j}} \right) \parallel \left( \frac{1}{|\mathcal{M}_t|} \sum_{M_k \in \mathcal{M}_t} h_{M_k} \right) \right]$ , where  $\parallel$  denotes concatenation. The actor network takes  $h_G$ , the operation and machine embeddings  $h_{O_{i,j}}$ ,  $h_{M_k}$ , and the pair-specific features  $h_{(O_{i,j}, M_k)}$  as input for the policy  $\pi_\psi$ , enabling it to select a machine–operation pair based on its embedding in relation to the global state.

The quantile critic in CDQAC uses a novel dueling architecture based on Wang et al. (2016), decomposing the state–action value into a value stream  $V(s)$  and an advantage stream  $A(s, a)$ . We adopt this as an architectural mitigation of the propagation problem that also motivates the delayed policy update: as the action space grows, only the selected pair’s advantage is updated per transition, so noisy critic estimates increasingly propagate to unsampled pairs through subsequent Bellman targets.  $V_\theta$ , by contrast, is updated at every step regardless of the sampled action, providing a stable shared state representation that anchors generalization across the action space and yields a more accurate  $Z_\theta$ . The two mechanisms are complementary: the delayed update stabilizes propagation temporally, the dueling decomposition stabilizes it architecturally. Whereas Wang et al. (2016) feed  $V_\theta$  and  $A_\theta$  the same input, we use separate inputs:  $V_\theta$  receives only  $h_G$ , while  $A_\theta$  additionally receives the operation-, machine-, and pair-specific embeddings (Fig. 2). This lets  $V_\theta$  focus on the state value while  $A_\theta$  specializes to each individual machine–operation pair  $(O_{i,j}, M_k)$ , resulting in the following formulation:

$$\begin{aligned} Z_\theta(h_{O_{i,j}}, h_{M_k}, h_{(O_{i,j}, M_k)}, h_G) &= V_\theta(h_G) + \left( A_\theta(h_{O_{i,j}}, h_{M_k}, h_{(O_{i,j}, M_k)}, h_G) \right. \\ &\quad \left. - \frac{1}{|\mathcal{A}(s_t)|} \sum_{(O', M') \in \mathcal{A}(s_t)} A_\theta(h_{O'}, h_{M'}, h_{(O', M')}, h_G) \right) \end{aligned} \quad (6)$$

where  $\mathcal{A}(s_t)$  is the set of available machine–operation pairs  $(O', M')$  in state  $s_t$ . In Eq. 6, we subtract the mean advantage from each action’s advantage, which is required because the value and advantage streams are not uniquely identifiable.

## 5 Experiments

**Generated & benchmark instances.** For both problems we train on 500 generated instances and evaluate on a mix of held-out generated instances and standard benchmarks. **FJSP:** we train on instances of sizes  $\{10 \times 5, 15 \times 10, 20 \times 10\}$ , where each job has between  $\lfloor 0.8m \rfloor$  and  $\lfloor 1.2m \rfloor$  operations ( $m$  being the number of machines) and processing times are integers drawn from  $[1, 99]$ . Evaluation uses 100 generated instances per size in  $\{10 \times 5, 15 \times 10, 20 \times 10, 30 \times 10, 40 \times 10\}$ , together with the Brandimarte (mk) (Brandimarte, 1993) and Hurink (edata, rdata, vdata) (Hurink et al., 1994) benchmarks. **JSP:** we train on 500 instances at  $10 \times 5$  and  $15 \times 10$  generated following Taillard (1993), and evaluate on the Taillard (Taillard, 1993) and Demirkol (Demirkol et al., 1998) benchmarks. Full benchmark details are given in App. C.

**Training dataset generation.** Offline RL trains on a fixed dataset  $D$ . We collect trajectories using three kind of heuristics: (i) Priority Dispatching Rules (**PDR**)—for FJSP, 4 job-selection  $\times$  4 machine-selection rules (16 trajectories per instance); for JSP, 4 job rules (machines fixed). (ii) Genetic Algorithms **GA** (Reijnen et al., 2023)—use the entire final population (typically higher quality, lower diversity than PDRs). (iii) **Random**—uniformly sample feasible actions. Consequently, we build four datasets: **PDR** (16 FJSP / 4 JSP trajectories), **GA** (200 trajectories per instance), **PDR-GA** (union), and **Random** (100 trajectories per instance). These datasets matches the setup used in offline RL work (Fu et al., 2020), where datasets with different qualities are used. From each trajectory, we extract the transitions on which CDQAC and offline RL baselines are trained. Heuristic details are in App. D.

**Metrics.** We report the *optimality gap*:  $\text{Gap} = \frac{C_{\max}^j - C_{\text{ub}}}{C_{\text{ub}}} \times 100$ , which measures the difference between  $C_{\max}^j$ , the makespan found by the method  $j$  and  $C_{\text{ub}}$ , which is the optimal or best-known makespan for the given instance. For generated instances, we used solutions generated by OR tools (Perron et al., 2023), with a solving time limit of 30 minutes per instance, as reported in Wang et al. (2023). For the benchmark instances, Taillard, Demirkol, Brandimarte, and Hurink, we used the best known solutions noted in the literature (Reijnen et al., 2023).

**Baselines.** We benchmark CDQAC against both offline and online RL approaches and strong heuristics. Each learning-based policy is evaluated in two modes: **greedy** (argmax) and **sampling** (100 solutions sampled; best kept), averaging over three evaluation seeds (1, 2, 3).

(1) **Offline RL:** We compare CDQAC with Offline-LD (Remmerden et al., 2025), Behavioral Cloning (BC), and Implicit Q-Learning (IQL) (Kostrikov et al., 2021). Offline-LD was originally developed for JSP and has not been applied to FJSP, IQL has not been applied to scheduling, and while BC has been used in related scheduling settings, it has not been evaluated in our setup. We therefore adapt all three baselines to FJSP using DAN (Wang et al., 2023) as the encoder network, matching CDQAC and isolating the contribution of its novel components. For Offline-LD, we additionally implement both the maskable QRDQN (mQRDQN) and discrete maskable SAC (d-mSAC) variants from (Remmerden et al., 2025). Each method is trained on the four datasets (PDR, GA, PDR-GA, Random) and three instance sizes ( $10 \times 5$ ,  $15 \times 10$ ,  $20 \times 10$ ), as relative offline RL performance can vary substantially between datasets. Full details are provided in App. E.

(2) **Online RL:** We benchmark CDQAC against state-of-the-art online RL methods: DANIEL (Wang et al., 2023) and Residual (Ho et al., 2024) because, similar to CDQAC, they can solve both JSP and FJSP instances directly. In addition, for FJSP, we also compare with FJSP-DRL (Song et al., 2023), and for JSP, L2D (Zhang et al., 2020) is added for comparison. We use published results of DANIEL and FJSP-DRL directly and retrain Residual on the same distribution (1,000 instances, 20 runs). We compare against GA from (Reijnen et al., 2023), dispatching rules (MOR-SPT/EST), CP (30 minutes time limit), and a specialized GA algorithm 2SGA (Defersha & Rooyani, 2020). Additional JSP baselines include MOR, MWKR, MIP, and CP, with a 30 minute time limit for both MIP and CP.

**Training Setup.** We evaluate the stability of CDQAC by running all experiments with four different seeds (1, 2, 3, 4). Although this is standard practice in offline RL (Fu et al., 2020), online RL methods for FJSP (Song et al., 2023; Wang et al., 2023) typically report results from a single seed. Consequently, we present mean and standard deviation for our offline RL comparisons, but only single seed results (seed 1) when comparing with online methods. Our hardware is a NVIDIA A100 GPU, Intel Xeon CPU, and 360GB of RAM. Detailed descriptions of the hyperparameters and the network architecture can be found in App. G.

## 5.1 Offline RL Methods Comparison with Different Training Datasets

Table 1 shows the results of our method compared to other offline approaches. CDQAC outperforms both versions of Offline-LD, IQL, and BC by a significant margin. Furthermore, CDQAC consistently outperforms all heuristics that generated the datasets ( $\pi_\beta$ ). In contrast, the other offline RL baselines, Offline-LD and IQL, performed poorly and never outperformed  $GA$  with the sampling evaluation, or even the PDR heuristics with greedy evaluation. The second highest performance was achieved with BC, when trained on  $GA$  (Greedy:13.91%, Sampling: 8.3%); however, BC still performed worse than CDQAC, even when trained

Table 1: Average gap (%) on all FJSP evaluation sets, for all training sizes ( $10 \times 5$ ,  $15 \times 10$ , and  $20 \times 10$ ).  $\pi_\beta$  best performance of heuristics that generated dataset. **Bold** is best result of the method (row) for each training dataset (column).

	PDR	GA	PDR-GA	Random	
Greedy	BC	29.13 $\pm$ 3.2	13.91 $\pm$ 0.6	22.37 $\pm$ 2.24	21.85 $\pm$ 2.51
	Offline-LD (mQRDQN)	22.26 $\pm$ 2.43	30.85 $\pm$ 3.57	21.80 $\pm$ 3.64	21.49 $\pm$ 2.62
	Offline-LD (d-mSAC)	23.28 $\pm$ 3.06	21.02 $\pm$ 2.13	25.94 $\pm$ 2.29	16.91 $\pm$ 1.89
	IQL	19.93 $\pm$ 1.83	20.66 $\pm$ 2.18	19.24 $\pm$ 2.34	21.34 $\pm$ 3.54
	<b>CDQAC (Ours)</b>	<b>12.34 <math>\pm</math> 1.72</b>	<b>13.06 <math>\pm</math> 2.10</b>	<b>11.31 <math>\pm</math> 1.33</b>	<b>10.68 <math>\pm</math> 0.51</b>
Sampling	BC	10.71 $\pm$ 0.99	8.3 $\pm$ 0.15	9.49 $\pm$ 0.56	13.15 $\pm$ 0.09
	Offline-LD (mQRDQN)	13.64 $\pm$ 0.20	14.26 $\pm$ 0.26	13.68 $\pm$ 0.17	13.63 $\pm$ 0.23
	Offline-LD (d-mSAC)	11.61 $\pm$ 1.32	8.83 $\pm$ 0.69	11.69 $\pm$ 1.23	7.79 $\pm$ 0.86
	IQL	10.01 $\pm$ 0.58	9.19 $\pm$ 0.56	9.48 $\pm$ 0.59	10.79 $\pm$ 0.74
	<b>CDQAC (Ours)</b>	<b>6.57 <math>\pm</math> 0.76</b>	<b>6.43 <math>\pm</math> 0.87</b>	<b>5.87 <math>\pm</math> 0.51</b>	<b>5.86 <math>\pm</math> 0.30</b>
$\pi_\beta$	14.13	6.74	6.74	28.16	

on the same *GA* dataset (Greedy: 13.06%, Sampling: 6.43%). Additional results of our offline RL comparison are in App. I.2.

Both Offline-LD (d-mSAC) and CDQAC achieve their best performance when trained on the Random dataset: Offline-LD (d-mSAC) reaches gaps of 16.91% and 7.79%, while CDQAC reaches 10.68% and 5.86% for greedy and sampling, respectively. For both methods, a diverse but suboptimal dataset is thus preferred over a narrow expert one, a pattern that differs from the data-quality ordering reported on standard offline-RL benchmarks (Schweighofer et al., 2022; Kumar et al., 2022). We explain this in detail in the following section.

## 5.2 Why scheduling structure favors broad-coverage data

Standard offline-RL benchmarks exhibit a clear data-quality ordering, in which random data is the weakest training source and noisy-expert data dominates (Schweighofer et al., 2022; Kumar et al., 2022). We argue that this ordering is not intrinsic to offline RL but follows from properties of the benchmarks on which it is typically evaluated. In case of scheduling, it has two structural properties. First, the reward  $r_t$  is the negative partial-makespan increase, so the undiscounted return equals the makespan: every transition carries a calibrated, objective-aligned signal rather than a sparse or proxy reward. Second, constructive scheduling policies share a fixed horizon—the number of operations—so random and expert heuristics produce equal-length trajectories, removing the trajectory-length bias that can favor expert data in other settings (Li et al., 2023). This allows an expert and random heuristic to explore the same states. Under these two conditions, performance hinges of offline learning for scheduling is not data *quality* but state–action *coverage*: a learner that estimates accurate returns and stitches high-value segments should prefer the dataset with the broadest coverage of the state–action space, even when every individual trajectory is poor. Since a random policy maximizes coverage by construction, we now empirically verify below that random data should be the *strongest*, source for scheduling due to the structural properties of scheduling.

Table 2: The **State-Action Coverage** (SACo) of the FJSP training datasets of each instance size. PDR is the reference dataset, and a higher SACo is better ( $\uparrow$ ).

Instance Size	PDR ( $\uparrow$ )	GA ( $\uparrow$ )	PDR-GA ( $\uparrow$ )	Random ( $\uparrow$ )
$10 \times 5$	1 $\pm$ 0	3.13 $\pm$ 0.38	4.13 $\pm$ 0.38	<b>8.46 <math>\pm</math> 0.71</b>
$15 \times 10$	1 $\pm$ 0	2.59 $\pm$ 0.46	3.59 $\pm$ 0.46	<b>6.93 <math>\pm</math> 0.29</b>
$20 \times 10$	1 $\pm$ 0	3.16 $\pm$ 0.4	4.16 $\pm$ 0.4	<b>7.7 <math>\pm</math> 0.18</b>
Average	1 $\pm$ 0	2.96 $\pm$ 0.49	3.96 $\pm$ 0.49	<b>7.7 <math>\pm</math> 0.77</b>

**Random data is the strongest source.** Table 1 shows the performance on different source datasets. As expected, CDQAC achieves its best gaps on the Random dataset (Greedy 10.68%, Sampling 5.86%) and on PDR-GA (11.31%, 5.87%), and its worst on the narrow PDR (12.34%, 6.57%) and expert GA (13.06%, 6.43%) datasets. The same ranking holds for Offline-LD (d-mSAC), which is also best on Random, indicating that the effect is a property of offline value learning on scheduling rather than of CDQAC specifically.

Table 3: Results on ablation study of dataset properties. Standard is the setup used in all other experiments. Trained on  $10 \times 5$ . **Bold**: indicates best gap (%).

	Training Dataset	Standard	Early Termination	Proxy Reward	Sparse Reward
Greedy	PDR	11.92 ± 0.49	<b>27.41 ± 3.88</b>	19.77 ± 5.02	13.73 ± 0.67
	GA	11.94 ± 0.38	38.04 ± 2.44	<b>17.6 ± 2.2</b>	<b>12.87 ± 0.82</b>
	PDR-GA	11.94 ± 0.53	33.78 ± 3.92	17.74 ± 3.31	12.89 ± 0.34
	Random	<b>11.61 ± 0.38</b>	39.11 ± 3.72	20.48 ± 3.32	13.74 ± 0.81
Sampling	PDR	<b>6.42 ± 0.18</b>	<b>13.14 ± 2.54</b>	10.06 ± 2.8	7.85 ± 0.53
	GA	6.58 ± 0.29	19.57 ± 3.05	9.66 ± 1.51	<b>6.68 ± 0.39</b>
	PDR-GA	6.57 ± 0.31	17.45 ± 2.49	<b>9.56 ± 2.05</b>	7.35 ± 0.2
	Random	6.43 ± 0.21	20.6 ± 3.1	11.41 ± 2.48	7.56 ± 0.51

**Coverage is the mechanism.** To make the coverage argument quantitative, we measure State-Action Coverage (SACo) (Schweighofer et al., 2022),  $\text{SACo}(D) = u_{s,a}(D)/u_{s,a}(D_{\text{ref}})$ , where  $u_{s,a}(D)$  denotes the number of unique state–action pairs in  $D$  and PDR is the reference dataset, so that  $\text{SACo}(\text{PDR}) = 1$ . Table 2 shows that Random has by far the highest coverage (on average  $7.7\times$  that of PDR), and the dataset ranking by SACo mirrors the performance ranking in Table 1. This is consistent with the prediction: for long-horizon problems ( $H \geq 40$ ), diverse coverage is known to support better policy learning than narrow expert data (Jin et al., 2021; Kumar et al., 2022), and FJSP has  $H \geq 50$  even at  $10 \times 5$ . Following Jin et al. (2021), wide coverage reduces the *intrinsic uncertainty*—the probability that transitions required by an optimal policy are absent from  $D$ —and enables CDQAC to confirm its pessimism (via CQL) rather than being forced toward a heuristic’s local optimum. App. F includes a more detailed explanation and additional experiments.

**Confirming the two preconditions.** As stated earlier, scheduling has unique structural properties: a dense, objective-aligned reward and equal-length trajectories. If either fails, the coverage advantage of random data should disappear. We test this through three ablations (Table 3): early termination (10% chance per timestep), proxy rewards (negative rewards adjusted to  $-1$ ), and sparse rewards (makespan given only at the terminal state). The predictions hold. Early termination breaks the equal-length property and is harmful on all datasets, and most so on Random: without access to later states, the intrinsic uncertainty is maximal and the coverage benefit vanishes, mirroring why random policies fail on Atari, where they terminate early at failure states (Li et al., 2023). Proxy rewards break objective alignment and degrade Random most relative to GA or PDR-GA, since stitching requires a reward that distinguishes good transitions from bad ones. With a sparse reward the degradation is smallest overall, but Random again degrades most, indicating that expert structure can partially substitute for a dense signal whereas coverage cannot. Together, these results confirm that this behavior is driven by exactly the two properties identified in advance, both of which the standard JSP and FJSP reward function (Zhang et al., 2020; Wang et al., 2023; Song et al., 2023; Iklassov et al., 2023; Remmerden et al., 2025; Smit et al., 2025) satisfies by construction—which places scheduling in a different regime from the settings where offline RL is usually studied, one where state–action coverage rather than data quality governs performance.

### 5.3 Comparison with online RL on FJSP

Having gained insights into CDQAC’s internal mechanisms, we now ask whether it is competitive with the dominant RL paradigm for both JSP and FJSP: online RL methods. Prior offline RL work (Fu et al., 2020; Fujimoto et al., 2019; Kumar et al., 2023) consistently reports that online methods dominate when both are available. For scheduling or COPs more broadly, Offline-LD is the only offline RL method reported to outperforms its online counterpart L2D (Remmerden et al., 2025), but it requires (near-)optimal CP solutions as training data. CDQAC, by contrast, is trained purely on random data without any online feedback from the environment or simulator. Therefore, achieving competitive results in this setting would represent a meaningful departure from established findings in offline RL.

Table 4 shows that CDQAC outperforms FJSP-DRL (Song et al., 2023), Residual (Ho et al., 2024), and DANIEL (Wang et al., 2023) on all benchmark sets, except the sampling evaluation of Hurink rdata, where DANIEL marginally edges out CDQAC (4.95% vs 5.59%). Because these benchmarks are drawn from distributions distinct from CDQAC’s training instances, this simultaneously demonstrates robustness to

Table 4: Results FJSP benchmarks sets. CDQAC trained on Random dataset; all models on  $10 \times 5$  or  $15 \times 10$  instances. **Bold** indicates best performance. The makespan and gap of 2SGA on la(vdata) are computed on instances la01–la30, as reported in Defersha & Rooyani (2020).

Method	mk		edata		rdata		vdata				
	Gap(%)	Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)			
Greedy	$10 \times 5$	FJSP-DRL	28.52	1.26	15.53	1.4	11.15	1.4	4.25	1.37	
		Residual	25.53	0.68	15.97	0.5	11.78	0.63	2.8	0.8	
		DANIEL	13.58	1.29	16.33	1.37	11.42	1.37	3.28	1.37	
		<b>CDQAC (Ours)</b>	13.04	1.1	<b>13.86</b>	1.18	<b>10.10</b>	1.18	<b>2.75</b>	1.18	
	$15 \times 10$	FJSP-DRL	26.77	1.25	15	1.4	11.14	1.4	4.02	1.37	
		Residual	25.22	0.68	16.99	0.5	11.19	0.62	4.04	0.79	
		DANIEL	12.97	1.3	14.41	1.38	12.07	1.36	3.75	1.37	
		<b>CDQAC (Ours)</b>	<b>12.64</b>	1.08	14.74	1.15	10.47	1.14	3.13	1.14	
	Sampling	$10 \times 5$	FJSP-DRL	18.56	4.13	8.17	4.91	5.57	4.81	1.32	4.71
			Residual	21.65	65.01	13.61	49.84	7.42	60.75	1.76	80.37
			DANIEL	9.53	4.12	9.08	4.71	<b>4.95</b>	4.73	0.69	4.77
			<b>CDQAC (Ours)</b>	8.96	3.36	9.4	3.82	5.59	3.84	<b>0.65</b>	3.84
$15 \times 10$		FJSP-DRL	19	4.13	8.69	4.87	5.95	4.82	1.34	4.72	
		Residual	19.91	66.09	11.94	50.61	8.25	61.52	1.58	77.59	
		DANIEL	8.95	4.08	8.72	4.7	5.49	4.73	0.72	4.75	
		<b>CDQAC (Ours)</b>	<b>7.94</b>	3.22	<b>7.77</b>	3.66	5.08	3.68	0.69	3.72	
MOR-SFT		25.67	0.1	17.75	0.11	14.38	0.1	6.06	0.11		
MOR-EST		29.59	0.1	17.59	0.11	14.3	0.1	5.59	0.11		
GA		14.29	232.95	4.55	237.06	4.43	243.91	0.67	283.97		
2SGA		3.17	57.7	-	-	-	-	0.39	51.43		
CP	1.5	1447	0	900	0.11	1397	0	639			

distributional shift. To our knowledge, this establishes CDQAC as the first offline RL method to outperform online RL on JSP/FJSP, while training purely on data from a random policy.

On generated instances (Table 5), CDQAC matches DANIEL at  $10 \times 5$  and outperforms it at  $15 \times 10$ , using only 500 training instances against the online baselines’ 1000. This pattern reflects two complementary factors, one on the problem side and one on the method side. On the problem side, the on-/off-policy distinction is central: DANIEL is trained with PPO and updates only from on-policy rollouts, so its policy fits the state distribution induced by the current policy on the training instance sizes, whereas CDQAC reuses every transition in  $\mathcal{D}$ , including those produced by random rollouts that lie well outside any on-policy distribution—and the properties identified in Sect. 5.2 (wide state–action coverage, a dense reward aligned with makespan, and equal-length trajectories) are exactly what make these off-distribution transitions informative rather than noise, unlike the failure-prone random data of typical offline-RL benchmarks. Online RL is therefore most competitive on evaluation sets that mirror its training distribution, while CDQAC’s relative advantage grows on out-of-distribution instances (Tables 4 and 6; significance analyses in App. J). On the method side, the quantile critic with its dueling architecture converts this coverage into an accurate return representation  $Z_\theta$  over  $\mathcal{D}$ , which the policy exploits through stitching to recover policies that surpass any individual trajectory in the data. The two factors are inseparable: coverage without an accurate critic cannot be stitched into a stronger policy, and an accurate critic without coverage has too few high-value segments to combine—a tension that becomes acute as the action space grows, as shown in Sect 5.5.

Direct training on large FJSP instances such as  $20 \times 10$  presents additional challenges due to the size of the action space, which grows from at most 50 machine-operation pairs at  $10 \times 5$  to 200 at  $20 \times 10$ . As the ablation in Sect. 5.5 shows, the delayed policy update and the dueling architecture become increasingly important as this action space grows, since only a single pair’s state–action value is updated per transition and any noisy estimate propagates to a larger fraction of subsequent Bellman targets—an effect that, unlike in online RL, cannot be corrected through environment interaction. App. H confirms that this is a training issue rather than a generalization issue, as CDQAC converges stably for both  $10 \times 5$  and  $15 \times 10$  across all training datasets, but not for  $20 \times 10$ . Nevertheless, our results in Table 6 demonstrate that training on large instance sizes is not required to achieve state-of-the-art performance. CDQAC learns scheduling policies from small problem sizes ( $10 \times 5$ ) that generalize successfully to larger problems; in fact, it outperforms DANIEL on large unseen instances (e.g.,  $30 \times 10$  and  $40 \times 10$  in Table 6), showing better generalization than online

Table 5: Results generated FJSP evaluation instances. CDQAC trained on Random dataset; training instances size is same as evaluation instance size. **Bold** indicates best performance per evaluation mode.

Method	10 × 5		15 × 10		20 × 10		
	Gap(%)	Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)	
Greedy	FJSP-DRL	16.03	0.45	16.33	1.43	10.15	1.91
	Residual	15.23	0.27	15.93	0.85	10.01	1.28
	DANIEL	<b>10.87</b>	0.45	12.42	1.35	<b>1.31</b>	1.85
	<b>CDQAC (Ours)</b>	11.56	0.39	<b>11.1</b>	1.16	4.34	1.56
Sampling	FJSP-DRL	9.66	1.11	12.13	3.98	9.64	6.23
	Residual	9.85	27.04	12.38	77.5	9.81	116.41
	DANIEL	<b>5.57</b>	0.74	6.79	3.89	<b>-1.03</b>	6.35
	<b>CDQAC (Ours)</b>	5.98	0.64	<b>5.85</b>	3.06	1.79	4.83
MOR-SPT	19.67	0.03	17.89	0.1	11.25	0.15	
MOR-EST	19.66	0.03	19.98	0.1	12.08	0.14	
GA	6.0	71.65	10.42	266.15	6.78	348.87	

Table 6: Generalization to large FJSP instances (30×10, and 40×10). CDQAC trained on Random dataset; training size 10×5. **Bold** indicates best performance per evaluation mode.

Method	30 × 10		40 × 10		
	Gap(%)	Time(s)	Gap(%)	Time(s)	
Greedy 10 × 5	FJSP-DRL	14.61	2.86	14.21	3.82
	Residual	13.16	2.11	12.82	3.1
	DANIEL	5.1	2.78	3.65	3.77
	<b>CDQAC (Ours)</b>	<b>4.43</b>	2.32	<b>3.17</b>	3.19
Sampling 10 × 5	FJSP-DRL	12.36	12.79	12.26	24.54
	Residual	12.94	213.89	12.85	319.69
	DANIEL	4.43	12.37	3.77	22.58
	<b>CDQAC (Ours)</b>	<b>3.11</b>	9.57	<b>2.21</b>	16.01
MOR-SPT	14.99	0.23	14.57	0.33	
MOR-EST	15.88	0.22	15.17	0.32	
GA	11.26	521.19	11.26	736.36	

Table 7: Results JSP benchmarks. Average gap (%) is reported. CDQAC trained on Random dataset for 10 × 5. For DANIEL Wang et al. (2023), only Taillard was reported. **Bold** indicates best result.

Instance Size	Greedy							Sampling			Exact		
	MWR	MOR	L2D	Offline-LD	DANIEL	Residual	<b>CDQAC (Ours)</b>	DANIEL	Residual	<b>CDQAC (Ours)</b>	MIP	CP	
Taillard	15 × 15	18.9	21.4	28.1	25.8	19.0	17.6	<b>15.0</b>	13.2	13.3	<b>10.4</b>	0.1	0.1
	20 × 15	23.0	23.6	32.7	30.2	22.1	21.2	<b>17.7</b>	17.4	16.1	<b>13.2</b>	3.2	0.2
	20 × 20	21.6	21.7	31.8	28.9	18.0	18.0	<b>17.6</b>	13.3	15.8	<b>12.9</b>	2.9	0.7
	30 × 15	24.3	23.2	30.2	29.2	21.7	20.1	<b>19.1</b>	17.2	18.0	<b>14.9</b>	10.7	2.1
	30 × 20	24.8	25.0	35.2	33.1	23.2	22.3	<b>21.2</b>	19.0	19.7	<b>17.9</b>	13.2	2.8
	50 × 15	16.5	17.3	21.0	20.6	14.8	15.6	<b>13.0</b>	12.7	13.2	<b>9.9</b>	12.2	3.0
	50 × 20	18.1	17.9	26.1	24.3	16.0	14.4	<b>12.8</b>	13.1	14.1	<b>11.0</b>	13.6	2.8
	100 × 20	8.3	9.1	13.3	12.7	7.3	6.5	<b>5.3</b>	5.9	6.5	<b>3.6</b>	11.0	3.9
	Mean	19.4	19.9	27.3	25.6	18.2	17.0	<b>15.2</b>	14.4	14.6	<b>11.7</b>	8.4	2.0
	Demirkol	20 × 15	27.8	30.3	36.3	35.8	–	26.1	<b>22.9</b>	–	22.6	<b>18.4</b>	5.3
20 × 20		26.8	26.9	34.4	32.8	–	21.5	<b>20.3</b>	–	18.9	<b>16.5</b>	4.7	1.9
30 × 15		31.9	36.4	37.8	38.8	–	27.6	<b>27.1</b>	–	29.4	<b>23.1</b>	14.2	2.5
30 × 20		31.9	33.7	38.0	36.0	–	29.9	<b>27.9</b>	–	28.3	<b>23.4</b>	16.7	4.4
40 × 15		26.5	35.5	34.6	35.5	–	26.2	<b>25.5</b>	–	28.4	<b>20.2</b>	16.3	4.1
40 × 20		32.0	35.9	39.2	38.5	–	27.7	<b>27.9</b>	–	30.9	<b>24.1</b>	22.5	4.6
50 × 15		27.3	34.8	33.2	34.1	–	27.4	<b>25.0</b>	–	29.5	<b>21.7</b>	14.9	3.8
50 × 20		29.9	36.5	37.7	38.9	–	30.0	<b>28.6</b>	–	32.8	<b>25.1</b>	22.5	4.8
Mean	29.2	33.7	36.4	36.3	–	27.0	<b>25.7</b>	–	27.6	<b>21.6</b>	14.6	3.5	

baselines. These results highlight the potential of offline RL, and directly tackling large-scale discrete action spaces—e.g., through factorized action spaces that decouple operation and machine selection—is an interesting future research direction.

#### 5.4 Comparison on JSP Instances

Table 7 shows the results for the JSP evaluation, demonstrating that CDQAC surpasses all baselines, including Offline-LD and online methods (Residual, DANIEL, L2D). Notably, CDQAC outperforms Offline-LD using only random data, despite the latter utilizing expert data. On Taillard instances, CDQAC achieves gaps of 15.2% (greedy) and 11.7% (sampling), beating DANIEL (18.2%/14.4%) and Residual (17.0%/14.6%). This advantage extends to the Demirkol benchmark, where CDQAC (25.7%/21.6% sampling) improves over Residual (27.0%/27.6%), confirming its effectiveness for JSP. Table 7 shows that CDQAC demonstrates favorable scaling on large Taillard instances. CDQAC’s sampling evaluation outperforms MIP on 50 × 15, and greedy on 50 × 20 and 100 × 20. For CP, CDQAC’s sampling evaluation outperforms on 100 × 20. This shows that CDQAC scales to larger JSP problem sizes more effectively than exact solvers, like MIP and CP. More results can be found in App. I.3 and in App. I.5, where we compare CDQAC to JSP-only baselines.

## 5.5 Ablation Study

We ablate each component by removing it and measuring the resulting average gap increase, on the Random dataset and across the three training sizes (Fig. 3; full per-size results in App. I.1). Two components dominate, and—critically—their importance grows sharply with the size of the action space. Removing the Delayed Policy Update increases the gap by 11.7% at  $10 \times 5$ , 38.5% at  $15 \times 10$ , and 112.8% at  $20 \times 10$  (54.3% on average), and removing our Dueling Critic with separated inputs for  $V_\theta$  and  $A_\theta$  follows the same trend on a smaller scale (4.3%  $\rightarrow$  5.8%  $\rightarrow$  16.7%, 8.9% on average). The Quantile Critic (4.2%) and CQL regularization (2.5%) contribute more modestly and roughly uniformly across sizes. The scale-dependence of the two dominant components is a sharp departure from online continuous control, where Fujimoto et al. (2019) report only an 11.4% degradation from removing delayed updates in MuJoCo and find delays beyond  $\eta = 2$  harmful; CDQAC instead achieves its best performance at  $\eta = 4$ .

**Sample Efficiency** We evaluated CDQAC’s sample efficiency by reducing the number of instances in the Random training dataset. Fig. 4 shows that increasing the size of the dataset has only a marginal positive effect on performance. We noticed the greatest performance difference for  $10 \times 5$  between 5 instances (greedy 11.8%) and 10 instances (greedy 10.5%), while other results show no significant differences. Due to CDQAC being an off-policy RL method, it is able to reuse examples effectively. Because the bootstrapped target  $\mathcal{TZ}(s, a) = r + \gamma Z_{\hat{\theta}}(s', a')$  with  $a' \sim \pi_\psi(\cdot | s')$  is recomputed against the current policy and target critic, the regression target for a given transition shifts as  $\pi_\psi$  and  $\hat{\theta}$  evolve, so each time a transition is resampled it carries a fresh learning signal rather than a redundant one. Therefore, CDQAC achieves near-full performance using only 1 to 5% of the original dataset. Extended results are in App. I.4.

## 6 Conclusion

We proposed CDQAC, a novel offline RL method for JSP and FJSP. CDQAC is an offline RL method for JSP, FJSP that has shown to outperform strong online RL baselines, while training purely on data from a random policy. This is a stronger setting than prior offline RL successes in scheduling, which required near-optimal CP solutions. CDQAC achieves this by learning an accurate representation of the returns of a possible scheduling action from a static dataset, enabling CDQAC to “stitch” together high-quality partial solutions into a new policy. A central message of this work is that the conventional offline-RL ordering—random data weakest, noisy-expert strongest—does not apply to scheduling problems. We argued from two structural properties of scheduling, a dense, objective-aligned reward and equal-length trajectories across heuristics, that coverage rather than data quality should govern offline performance, and therefore that random data should be the strongest source; our experiments confirm both the prediction and the coverage mechanism behind it. CDQAC also generalizes well from small to larger instance sizes.

Offline RL remains underexplored in scheduling and, more broadly, in COPs. In future work, we plan to extend our approach to other combinatorial optimization problems, and to real-world scheduling problems, for which building a simulated environment is infeasible but has suboptimal training data generated by heuristics.

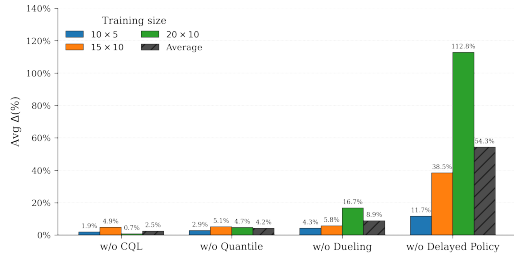


Figure 3: The results of the ablation study on each component of CDQAC: CQL, Quantile Critic, Dueling architecture, and Policy Delay. The results show the average increase in gap  $\Delta(\%)$  for each training size. Lower is better.

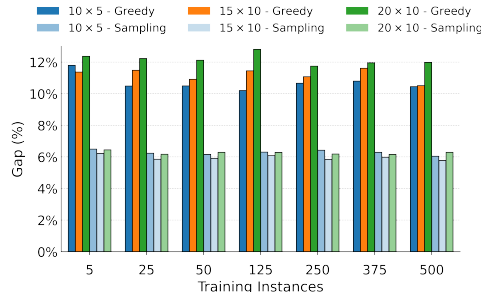


Figure 4: Results of reducing the number of instances in Random dataset, evaluated on FJSP benchmarks Brandimarte and Hurink.

## References

- An, G., Moon, S., Kim, J.-H., and Song, H. O. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In *Neural Information Processing Systems*, 2021.
- Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 449–458. JMLR.org, 2017.
- Bhatt, N. and Chauhan, N. R. Genetic algorithm applications on job shop scheduling problem: A review. In *2015 International Conference on Soft Computing Techniques and Implementations (ICSCTI)*, pp. 7–14, 2015. doi: 10.1109/ICSCTI.2015.7489556.
- Brandimarte, P. Routing and scheduling in a flexible job shop by tabu search. *Ann. Oper. Res.*, 41(1–4): 157–183, May 1993. ISSN 0254-5330.
- Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 15084–15097. Curran Associates, Inc., 2021. URL [https://proceedings.neurips.cc/paper\\_files/paper/2021/file/7f489f642a0ddb10272b5c31057f0663-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2021/file/7f489f642a0ddb10272b5c31057f0663-Paper.pdf).
- Christodoulou, P. Soft actor-critic for discrete action settings. *CoRR*, abs/1910.07207, 2019. URL <http://arxiv.org/abs/1910.07207>.
- Corsini, A., Porrello, A., Calderara, S., and Dell’Amico, M. Self-labeling the job shop scheduling problem. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=buqvMT3B4k>.
- Da Col, G. and Teppan, E. C. Industrial-size job shop scheduling with constraint programming. *Operations Research Perspectives*, 9:100249, 2022. ISSN 2214-7160.
- Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. Distributional reinforcement learning with quantile regression. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI’18/IAAI’18/EAAI’18. AAAI Press, 2018. ISBN 978-1-57735-800-8.
- Defersha, F. M. and Rooyani, D. An efficient two-stage genetic algorithm for a flexible job-shop scheduling problem with sequence dependent attached/detached setup, machine release date and lag-time. *Computers & Industrial Engineering*, 147:106605, 2020. ISSN 0360-8352. doi: <https://doi.org/10.1016/j.cie.2020.106605>. URL <https://www.sciencedirect.com/science/article/pii/S0360835220303399>.
- Demirkol, E., Mehta, S., and Uzsoy, R. Benchmarks for shop scheduling problems. *European Journal of Operational Research*, 109(1):137–141, 1998. ISSN 0377-2217. doi: [https://doi.org/10.1016/S0377-2217\(97\)00019-2](https://doi.org/10.1016/S0377-2217(97)00019-2). URL <https://www.sciencedirect.com/science/article/pii/S0377221797000192>.
- Drakulic, D., Michel, S., Mai, F., Sors, A., and Andreoli, J.-M. Bq-nco: bisimulation quotienting for efficient neural combinatorial optimization. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS ’23, Red Hook, NY, USA, 2023. Curran Associates Inc.
- Fan, H. and Su, R. Mathematical modelling and heuristic approaches to job-shop scheduling problem with conveyor-based continuous flow transporters. *Computers & Operations Research*, 148:105998, 2022. ISSN 0305-0548. doi: <https://doi.org/10.1016/j.cor.2022.105998>. URL <https://www.sciencedirect.com/science/article/pii/S0305054822002313>.
- Figueiredo Prudencio, R., Maximo, M. R. O. A., and Colombini, E. L. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 35(8):10237–10257, 2024. doi: 10.1109/TNNLS.2023.3250269.

- Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning, 2020.
- Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.
- Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pp. 2052–2062, 2019.
- Ho, K.-H., Cheng, J.-Y., Wu, J.-H., Chiang, F., Chen, Y.-C., Wu, Y.-Y., and Wu, I.-C. Residual scheduling: A new reinforcement learning approach to solving job shop scheduling problem. *IEEE Access*, 12:14703–14718, 2024. doi: 10.1109/ACCESS.2024.3357969.
- Hurink, J., Jurisch, B., and Thole, M. Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum*, 15(4):205–215, Dec 1994. ISSN 1436-6304. doi: 10.1007/BF01719451. URL <https://doi.org/10.1007/BF01719451>.
- Iklassov, Z., Medvedev, D., De Retana, R. S. O., and Takac, M. On the study of curriculum learning for inferring dispatching policies on the job shop scheduling. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI '23, 2023. ISBN 978-1-956792-03-4. doi: 10.24963/ijcai.2023/594. URL <https://doi.org/10.24963/ijcai.2023/594>.
- Jackson, M. T., Berdica, U., Liesen, J. L., Whiteson, S., and Foerster, J. N. A clean slate for offline reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL <https://openreview.net/forum?id=8P3QNSckMp>.
- Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 1273–1286. Curran Associates, Inc., 2021. URL [https://proceedings.neurips.cc/paper\\_files/paper/2021/file/099fe6b0b444c23836c4a5d07346082b-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2021/file/099fe6b0b444c23836c4a5d07346082b-Paper.pdf).
- Jin, Y., Yang, Z., and Wang, Z. Is pessimism provably efficient for offline rl? In *International conference on machine learning*, pp. 5084–5096. PMLR, 2021.
- Kingma, D. P. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. *CoRR*, abs/2110.06169, 2021. URL <https://arxiv.org/abs/2110.06169>.
- Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.
- Kumar, A., Hong, J., Singh, A., and Levine, S. Should i run offline reinforcement learning or behavioral cloning? In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=AP1MKT37rJ>.
- Kumar, A., Agarwal, R., Geng, X., Tucker, G., and Levine, S. Offline q-learning on diverse multi-task data both scales and generalizes. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=4-k7kUavAj>.
- Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020. URL <https://arxiv.org/abs/2005.01643>.
- Li, A., Misra, D., Kolobov, A., and Cheng, C.-A. Survival instinct in offline reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=shePL2nbwl>.

- Luo, F., Lin, X., Liu, F., Zhang, Q., and Wang, Z. Neural combinatorial optimization with heavy decoder: toward large scale generalization. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.
- Ma, Y. J., Jayaraman, D., and Bastani, O. Conservative offline distributional reinforcement learning. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL <https://openreview.net/forum?id=Z2vksUFuVst>.
- Park, J., Bakhtiyar, S., and Park, J. Schedulenet: Learn to solve multi-agent scheduling problems with reinforcement learning. *arXiv preprint arXiv:2106.03051*, 2021.
- Perron, L., Didier, F., and Gay, S. The cp-sat-lp solver. In Yap, R. H. C. (ed.), *29th International Conference on Principles and Practice of Constraint Programming (CP 2023)*, volume 280 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 3:1–3:2, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-300-3. doi: 10.4230/LIPIcs.CP.2023.3. URL <https://drops.dagstuhl.de/opus/volltexte/2023/19040>.
- Pirnay, J. and Grimm, D. G. Self-improvement for neural combinatorial optimization: Sample without replacement, but improvement. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=agT8ojoHOX>. Featured Certification.
- Reijnen, R., van Straaten, K., Bukhsh, Z., and Zhang, Y. Job shop scheduling benchmark: Environments and instances for learning and non-learning methods. *arXiv preprint arXiv:2308.12794*, 2023.
- Remmerden, J. v., Bukhsh, Z., and Zhang, Y. Offline reinforcement learning for learning to dispatch for job shop scheduling. *Machine Learning*, 114(8):191, 2025. doi: 10.1007/s10994-025-06826-w. URL <https://doi.org/10.1007/s10994-025-06826-w>.
- Schweighofer, K., Dinu, M.-c., Radler, A., Hofmarcher, M., Patil, V. P., Bitto-Nemling, A., Eghbal-zadeh, H., and Hochreiter, S. A dataset perspective on offline reinforcement learning. In *Conference on Lifelong Learning Agents*, pp. 470–517. PMLR, 2022.
- Smit, I. G., Zhou, J., Reijnen, R., Wu, Y., Chen, J., Zhang, C., Bukhsh, Z., Zhang, Y., and Nuijten, W. Graph neural networks for job shop scheduling problems: A survey. *Comput. Oper. Res.*, 176(C), April 2025. ISSN 0305-0548. doi: 10.1016/j.cor.2024.106914. URL <https://doi.org/10.1016/j.cor.2024.106914>.
- Song, W., Chen, X., Li, Q., and Cao, Z. Flexible job-shop scheduling via graph neural network and deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, 19(2):1600–1610, 2023. doi: 10.1109/TII.2022.3189725.
- Taillard, E. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2): 278–285, 1993. ISSN 0377-2217. doi: [https://doi.org/10.1016/0377-2217\(93\)90182-M](https://doi.org/10.1016/0377-2217(93)90182-M). URL <https://www.sciencedirect.com/science/article/pii/037722179390182M>. Project Management and Scheduling.
- Tassel, P., Gebser, M., and Schekotihin, K. An end-to-end reinforcement learning approach for job-shop scheduling problems based on constraint programming. In *Proceedings of the Thirty-Third International Conference on Automated Planning and Scheduling*, ICAPS '23. AAAI Press, 2023. ISBN 1-57735-881-3. doi: 10.1609/icaps.v33i1.27243. URL <https://doi.org/10.1609/icaps.v33i1.27243>.
- Veronique Sels, N. G. and Vanhoucke, M. A comparison of priority rules for the job shop scheduling problem under different flow time- and tardiness-related objective functions. *International Journal of Production Research*, 50(15):4255–4270, 2012.
- Wang, R., Wang, G., Sun, J., Deng, F., and Chen, J. Flexible job shop scheduling via dual attention network-based reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2023. doi: 10.1109/TNNLS.2023.3306421.
- Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., and De Freitas, N. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 1995–2003. JMLR.org, 2016.

- Zhang, C., Song, W., Cao, Z., Zhang, J., Tan, P. S., and Xu, C. Learning to dispatch for job shop scheduling via deep reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.
- Zhang, C., Cao, Z., Song, W., Wu, Y., and Zhang, J. Deep reinforcement learning guided improvement heuristic for job shop scheduling. In *The Twelfth International Conference on Learning Representations*, 2024a.
- Zhang, C., Cao, Z., Wu, Y., Song, W., and Sun, J. Learning topological representations with bidirectional graph attention network for solving job shop scheduling problem. In *Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence*, 2024b.
- Zhou, H., Wei, T., Lin, Z., Li, J., Xing, J., Shi, Y., Shen, L., Yu, C., and Ye, D. Revisiting discrete soft actor-critic. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=EUf2R6VBeU>.

**Algorithm 1** Training Procedure of CDQAC

---

**Input:** Dataset  $D$ , batch size  $B$ , policy update frequency  $\eta$ , total training steps  $T$ , CQL coefficient  $\alpha_{\text{CQL}}$ , entropy coefficient  $\lambda$ , target update rate  $\rho$ , learning rates  $\ell_\psi, \ell_\theta$   
Initialize policy network  $\psi$ , critic network  $\theta$ , target network  $\hat{\theta} \leftarrow \theta$   
**for**  $t = 1$  **to**  $T$  **do**  
  Sample mini-batch  $\{(s_i, a_i, r_i, s'_i)\}_{i=1}^B \sim D$   
  Compute target quantiles:  $\mathcal{T}Z_i \leftarrow r_i + \gamma Z_{\hat{\theta}}(s'_i, a'_i)$  where  $a'_i \sim \pi_\psi(\cdot | s'_i)$   
  Compute TD loss:  $\mathcal{L}_{\text{TD}}(\theta) \leftarrow \frac{1}{B} \sum_{i=1}^B \sum_{j=1}^N \rho_{\tau_j}^H (\mathcal{T}Z_i - Z_\theta(s_i, a_i))$   
  Compute conservative critic loss:  $\mathcal{L}_Z(\theta) \leftarrow \frac{1}{B} \sum_{i=1}^B \left[ \log \sum_{a' \in \mathcal{A}(s_i)} \exp(Q_\theta^Z(s_i, a')) - Q_\theta^Z(s_i, a_i) \right] + \mathcal{L}_{\text{TD}}(\theta)$   
  Update critic:  $\theta \leftarrow \theta + \ell_\theta \nabla_\theta \mathcal{L}_Z(\theta)$   
  **if**  $t \bmod \eta = 0$  **then**  
    Compute policy loss:  $\mathcal{L}_\pi(\psi) \leftarrow \frac{1}{B} \sum_{i=1}^B \left[ \sum_{a \in \mathcal{A}(s_i)} -Q_\theta^Z(s_i, a) \pi_\psi(a | s_i) + \lambda \mathcal{H}[\pi_\psi(\cdot | s_i)] \right]$   
    Update policy:  $\psi \leftarrow \psi + \ell_\psi \nabla_\psi \mathcal{L}_\pi(\psi)$   
  **end if**  
  Update target network:  $\hat{\theta} \leftarrow (1 - \rho)\hat{\theta} + \rho\theta$   
**end for**

---

**A Pseudocode**

Algorithm 1 shows the training process of CDQAC. In it, we train CDQAC using a static dataset  $D = (s, a, r, s')$  of scheduling transitions. At each training step, we sample a mini-batch of  $B$  transitions from  $D$ . For each transition, we compute the target  $\mathcal{T}Z = r + \gamma Z_{\hat{\theta}}(s', a')$  using the target network  $\hat{\theta}$  and next actions  $a' \sim \pi_\psi(\cdot | s')$  drawn from the current policy. The critic is optimized through a conservative quantile-based objective, combining the temporal difference (TD) loss  $\mathcal{L}_{\text{TD}}$  (Eq. 3) with a CQL penalty that discourages overestimation of out-of-distribution actions (Eq. 4). The critic parameters  $\theta$  are updated via gradient descent on the combined loss  $\mathcal{L}_Z$ .

To stabilize training, we employ a delayed policy update strategy: the actor  $\pi_\psi$  is updated every  $\eta$  steps by minimizing the Q-learning objective (Eq. 5), with the entropy bonus  $\mathcal{H}[\pi_\psi(\cdot | s)]$ . The policy update relies on the scalarized quantile values  $Q_\theta^Z(s, a) = \mathbb{E}[Z_\theta(s, a)]$ , where  $Z_\theta$  is the minimum of two dueling quantile networks. Finally, the target network is updated using Polyak averaging:  $\hat{\theta} \leftarrow (1 - \rho)\hat{\theta} + \rho\theta$ .

**B Network Architecture**

The dual attention network (Wang et al., 2023) (DAN) is an attention-based network architecture for JSP and FJSP that encodes the operation features  $h_{O_{i,j}}^{(L)}$ , and machine features  $h_{M_k}^{(L)}$ , where  $L$  presents the current layer input, so  $L = 1$  is the input features. DAN is able to learn the complex relation between each operation  $O_{i,j}$  and each compatible machine  $M_k$ , through separate *operation attention blocks* and *machine attention blocks* as seen in Fig. 2 in Sect. 4.1. In this section, we provide an overview of each attention block, and their interaction. Afterwards, we state the features used for the operations, machines and machine-operation pairs.

**Operation Attention Block.** To capture the sequential nature of operations within jobs, the operation attention blocks attend each operation  $O_{i,j}$  in the context of its predecessor  $O_{i,j-1}$  and successor  $O_{i,j+1}$ , if they exist. An attention coefficient is calculated between these operations:

$$a_{i,j,p} = \text{Softmax} \left( \text{LeakyReLU} \left( \mathbf{V}^T \left[ \left( \mathbf{W} h_{O_{i,j}}^{(L)} \parallel \mathbf{W} h_{O_{i,p}}^{(L)} \right) \right] \right) \right), \quad (7)$$

where  $\mathbf{W}$ , and  $\mathbf{V}$  are learned projections. The attention coefficient  $a_{i,j,p}$ , calculated in Eq. 7, is used to calculate the output of the operation attention block as follows:

$$h_{O_{i,j}}^{(L+1)} = \sigma \left( \sum_{p=j-1}^{j+1} a_{i,j,p} \mathbf{W} h_{O_{i,p}}^{(L)} \right), \quad (8)$$

where  $\sigma$  is an activation function. The operation blocks in DAN (Wang et al., 2023) function similar to a GNN, in that information, one by one, is propagated through the operations.

**Machine Attention Block.** The machine attention block considers the relationship between two machines  $M_y \in \mathcal{M}_t$  and  $M_z \in \mathcal{M}_t$  in relation to the set of unscheduled operations  $\hat{O}_{y,z}$  that can be processed by either  $M_y$  or  $M_z$ . The embedding of the pooled operation is calculated as  $h_{\hat{O}_{y,z}}^{(L)} = \frac{1}{|\hat{O}_{y,z}|} \sum_{O_{i,j} \in \hat{O}_{y,z} \cap \mathcal{O}_c} h_{O_{i,j}}^{(L)}$ , where  $\mathcal{O}_c$  represents the current operations available to schedule. The attention in this block is calculated through:

$$u_{y,z} = \text{Softmax}\left(\text{LeakyReLU}\left(\mathbf{X} \left[ (\mathbf{Y}h_{M_y}^{(L)}) \parallel (\mathbf{Y}h_{M_z}^{(L)}) \parallel (\mathbf{Z}h_{\hat{O}_{y,z}}^{(L)}) \right]\right)\right) \quad (9)$$

where  $\mathbf{X}$ ,  $\mathbf{Y}$ , and  $\mathbf{Z}$  are linear projections. Whenever two machines  $M_y$  and  $M_z$  do not share any operations in the current candidate set  $\hat{O}_{y,z} \cap \mathcal{J}_c = \emptyset$ , we set the attention  $u_{y,z}$  to zero. The output of the machine operation block is calculated as:

$$h_{M_k}^{(L+1)} = \sigma \left( \sum_{q \in \mathcal{N}_k} u_{k,q} \mathbf{Y}h_{M_q}^{(L)} \right), \quad (10)$$

where  $\mathcal{N}_k$  is the set of machines, for which  $M_k$  shares operations, including  $M_k$  itself.

Lastly, DAN (Wang et al., 2023) uses a multihead attention approach, whereby each operation attention and machine attention block consist of  $H$  heads. The results of the  $H$  heads can be concatenated or averaged. Following the prior work of Wang et al. (2023), we concatenate the heads for each layer, except the last layer, which was averaged over the  $H$  heads. We use ELU as our activation function for both operation and machine attention blocks.

## B.1 Features

Table 8 shows the features used in our paper, based on the prior work of Wang et al. (2023). Both the machine features  $M_k$  and the operation features  $O_{i,j}$  are embedded using the DAN network. These embeddings, with the machine-operation pair  $(O_{i,j}, M_k)$  features are used as input for the quantile critic and actor networks. In Table 8, we introduce the notation  $\mathcal{O}_k$ , which represents all operations  $O_{i,j} \in \mathcal{O}_k$  that  $M_k$  can process.

## C Benchmark Instance Sets

As described in Sect.5, we evaluate our approach on generated instance sets as well as four established benchmark sets. For FJSP, we use the generated evaluation instances, the Brandimarte (mk) benchmark (Brandimarte, 1993) and the Hurink benchmark (Hurink et al., 1994), which includes the edata, rdata, and vdata subsets. For JSP, we evaluate on the Taillard (Taillard, 1993) and Demirkol (Demirkol et al., 1998) benchmarks. For each benchmark, we report the range of processing times, number of jobs, number of machines, and, specifically for FJSP, the number of machines available per operation.

### C.1 FJSP

**Generated Evaluation Instances.** We generated 100 instances for each of the following sizes:  $10 \times 5$ ,  $15 \times 10$ ,  $20 \times 10$ ,  $30 \times 10$ ,  $40 \times 10$ , using the same generation procedure as for the training data (Sect. 5). Each operation is assigned between 1 and  $|\mathcal{M}|$  available machines, selected uniformly at random.

**Brandimarte (mk) Benchmark.** The Brandimarte benchmark (Brandimarte, 1993) comprises 10 instances, each with 10 to 20 jobs and 4 to 15 machines. Processing times range from 1 to 19. The average number of machines available per operation ranges from 1.4 to 4.1, depending on the instance.

**Hurink Benchmark.** The Hurink benchmark (Hurink et al., 1994) consists of three subsets, edata, rdata, and vdata, each containing 40 instances. These subsets vary in degree of flexibility, with edata providing the lowest and vdata the highest average number of machines per operation. All instances include between 7 and

Table 8: Features used by CDQAC, separated by operation  $O_{i,j}$ , machine  $M_k$ , and machine-operation pair  $(O_{i,j}, M_k)$ .

Feature	Description
Operation Features $O_{i,j}$	
Min. proc. time	$\min_{M_k \in \mathcal{M}_{i,j}} p_{i,j}^k$
Mean proc. time	$\frac{1}{ \mathcal{M}_{i,j} } \sum_{M_k \in \mathcal{M}_{i,j}} p_{i,j}^k$
Span proc. time	$\max_{M_k \in \mathcal{M}_{i,j}} p_{i,j}^k - \min_{M_k \in \mathcal{M}_{i,j}} p_{i,j}^k$
Compatibility ratio	$\frac{ \mathcal{M}_{i,j} }{ \mathcal{M} }$
Scheduled	1 if scheduled, 0 otherwise
Estimated LB	Estimated lower bound completion time $C(O_{i,j})$
Remaining ops $J_i$	Number of unscheduled operations in $J_i$
Remaining proc. time $J_i$	Total proc. time of unscheduled operations in $J_i$
Waiting time	Time since $O_{i,j}$ became available
Remaining proc. time	Remaining processing time (0 if not started)
Machine Features $M_k$	
Min. proc. time	$\min_{O_{i,j} \in \mathcal{O}_k} p_{i,j}^k$
Mean proc. time	$\frac{1}{ \mathcal{O}_k } \sum_{O_{i,j} \in \mathcal{O}_k} p_{i,j}^k$
Total unscheduled ops	$ \mathcal{O}_k $
Schedulable ops at $t$	# of ops schedulable at timestep $t$
Free time	Time until $M_k$ becomes available
Waiting time	0 if $M_k$ is working
Working status	1 if working, 0 otherwise
Remaining proc. time	Time left on current task (0 if idle)
Machine-Operation Pair $(O_{i,j}, M_k)$	
Processing time	$p_{i,j}^k$
Ratio to max of $O_{i,j}$	$\frac{p_{i,j}^k}{\max_{M_k} p_{i,j}^k}$
Ratio to max schedulable on $M_k$	$\frac{p_{i,j}^k}{\max_{p_{i,j}^k \in \mathcal{O}_k(t)} p_{i,j}^k}$
Ratio to global max	$\frac{p_{i,j}^k}{\max_{p_{i,j}^k \in \mathcal{O}} p_{i,j}^k}$
Ratio to $M_k$ 's unscheduled max	$\frac{p_{i,j}^k}{\max_{p_{i,j}^k \in \mathcal{O}_k} p_{i,j}^k}$
Ratio to compatible max	$\frac{p_{i,j}^k}{\max_{p_{i,j}^k \in \mathcal{M}_{i,j}} p_{i,j}^k}$
Ratio to $J_i$ workload	$\frac{p_{i,j}^k}{\sum_{p_{i,j}^k \in J_i} p_{i,j}^k}$
Joint waiting time	Sum of $O_{i,j}$ and $M_k$ waiting times

30 jobs and between 4 and 15 machines, with processing times between 5 and 99. The average number of machines available per operation is as follows: **edata**: Between 1.13 and 1.2, **rdata**: Between 1.88 and 2.06, and **vdata**: Between 2.38 and 6.7.

## C.2 JSP

**Taillard Benchmark.** The Taillard benchmark (Taillard, 1993) contains 80 instances, ranging from  $15 \times 15$  to  $100 \times 20$ . Processing times range between 1 and 99. These instances are similar to those used to train CDQAC.

**Demirkol Benchmark.** The Demirkol benchmark (Demirkol et al., 1998) includes 80 instances, with instance sizes ranging from  $20 \times 15$  to  $80 \times 20$ . Processing times range from 1 to 200, twice the maximum value found in Taillard and CDQAC's training data.

## D Details of Dataset Generation Heuristics

Our experimental setup in Sect. 5 stated that we used three types of heuristics to generate our training datasets, namely, priority dispatching rules (PDR), genetic algorithms (GA) and a random policy. We will now give a detailed explanation of each heuristic, and, in the case of GA, the hyperparameters.

### D.1 Priority Dispatching Rules (PDR)

For the priority dispatching rules (PDR), we have separate rules for the selection of *jobs* and *machines* for FJSP. In our setup, first, a job  $J_i \in \mathcal{J}$  is selected by the job selection rule. This job selection rule selects a job based on a specific rule, in which it is checked if there are still operations in  $J_i$  to be scheduled. The machine selection rule selects the machine  $M_k \in \mathcal{M}_{i,j}$  for operation  $O_{i,j} \in J_i$ , where  $O_{i,j}$  is the current operation in  $J_i$  that needs to be scheduled. For JSP, we only considered the job selection rules, since only one machine is ever available per operation. Furthermore, both the job and machine selection rules follow the MDP formulation, stated in Sect. 2, by which operation  $O_{i,j}$  can only be scheduled on  $M_k$ , if it is free at timestep  $t$ . In the following, we give an overview of the job selection rules and the machine selection rules.

**Job selection rules.** We utilized four different job selection rules, namely, *Most Operations Remaining* (MOR), *Least Operations Remaining* (LOR), *Most Work Remaining* (MWR), and *Least Work Remaining* (LWR). Both MOR and LOR decide on the basis of the number of unscheduled operations in a job  $J_i$ . MOR selects the job with the most operations and LOR selects the job with the least operations to be scheduled. MWR and LWR focus on the remaining total processing times, a.k.a. the summation of processing times in a  $J_i$ , whereby we average the processing times of the available machines  $M_k \in \mathcal{M}_{i,j}$ . MWR selects the job with the highest total remaining processing times, whereas LWR selects the job with the least.

**Machine selection rules.** We considered four different machine selection rules, namely, *Shortest Processing Time* (SPT), *Longest Processing Time* (LPT), *Earliest Start Time* (EST), and *Latest Start Time* (LST). Both SPT and LPT select a machine  $M_k \in \mathcal{M}_{i,j}$  for operation  $O_{i,j}$  based on the processing time, with SPT selecting the machine with the shortest and LPT with the longest. EST and LST consider how long a machine  $M_k$  is already free, with EST selecting the machine that is free the shortest, and LST the longest.

### D.2 Genetic Algorithms (GA)

For our genetic algorithm (GA), we used the implementation of Reijnen et al. (2023), whereby we introduced the constraint that  $O_{i,j}$  can only be scheduled if machine  $M_k$  is free at that time. This results in a more tight solution, with no gaps. Furthermore, we used a population size of 200, and ran the GA for 100 generations. The crossover probability was set at 0.7, and the mutation probability at 0.2.

### D.3 Random Policy

The random policy adheres to the MDP introduced in Sect. 2. This means that the random policy selects a random machine-operation pair based on those available at the time step  $t$ . The random policy can only select a machine-operation pair, if it can be scheduled at timestep  $t$ .

## E Details of Offline Reinforcement Learning Baselines

For our comparison of CDQAC to Offline-LD (Remmerden et al., 2025) in Sect. 5.1, we adapted both versions of it, namely, Offline-LD with a maskable Quantile Regression DQN (mQRDQN) and with a discrete maskable Soft Actor-Critic (d-mSAC), using a dual attention network (Wang et al., 2023), such that both versions of Offline-LD used the same encoding as our introduced CDQAC approach. We provide a brief explanation of our implementations of each method, in which we state the hyperparameters used for each. If a hyperparameter is not stated, it is the same as CDQAC, as stated in App. G.

**Offline-LD (mQRDQN).** The mQRDQN version of Offline-LD is implemented identically as described by Remmerden et al. (2025). The hyperparameters are identical to CDQAC, whereby we set  $\ell_\theta = 2 \times 10^{-4}$ . In the original implemented of Offline-LD (mQRDQN) was not able to sample actions; therefore, for the sampling evaluation, we use Boltzmann sampling.

**Offline-LD (d-mSAC).** For d-mSAC version of Offline-LD, we implemented both the policy network and the Q network with a separate dual attention network (Wang et al., 2023) for each. We used the

hyperparameters as with CDQAC, except for  $\alpha_{CQL}$ , which we set to  $\alpha_{CQL} = 0.1$ , and the target entropy of d-mSAC, which we set to 0.3. During initial testing, we found that this increased stability and performance with d-mSAC.

**Implicit Q-learning.** The main difference between Implicit Q-learning (IQL) (Kostrikov et al., 2021) and Offline-LD and CDQAC is that IQL constrains training by not using OOD actions, whereas Offline-LD and CDQAC regularize the Q-values of OOD actions during training to prevent overestimation. IQL consists of three networks, a policy, a value, and a Q network. Two hyperparameters of IQL are important to mention, namely  $\beta_{IQL}$  and  $\tau_{IQL}$ . Firstly,  $\beta_{IQL}$  controls how much the policy should learn to "exploit" the learned Q-values, or if it should stay close to the behavior found in the dataset, with  $\beta_{IQL} = 0$ , being equal to behavioral cloning. We decided, due to the suboptimality of our training datasets, to set  $\beta_{IQL} = 15$ .  $\tau_{IQL}$  controls how much IQL should focus on positive examples, whereby  $\tau_{IQL} = 0.5$  is equal to a SARSA update. Kostrikov et al. (2021) reported settings between 0.7 and 0.9 for  $\tau_{IQL}$ . We therefore tested 0.7, 0.8 and 0.9 to identify the ideal value and found that  $\tau_{IQL} = 0.7$  result in the most stable updates. We set all learning rates at  $2 \times 10^{-4}$ , by which we also tested  $2 \times 10^{-5}$ ; however, we found that this did not produce good results.

**Behavioral Cloning** Behavioral Cloning (BC) learns to imitate the behavioral policy  $\pi_\beta$ , which generated the training dataset. The BC loss is the cross-entropy loss between the predicted action for each state and the action found in the dataset. BC only trains a policy network and does not use a critic. All hyperparameters are the same as CDQAC (Table 10).

## F Expanded Explanation of Results

In Sect. 5, we showed that CDQAC achieved the highest performance with the dataset generated with a random heuristic. In this appendix, we further state theoretical and empirical properties enabling CDQAC to achieve the performance when trained on suboptimal random datasets. We analyze this through State-Action Coverage (SACo), Intrinsic Uncertainty, and Trajectory Stitching.

### F.1 State Action Coverage and Pessimism

In Table 2, we show the coverage of each dataset. This entails the following properties of each dataset:

- **PDR:** A narrow dataset of deterministic dispatching policies, which explore a small section of the possible state action pairs  $(s, a)$ .
- **GA:** A wider coverage compared to PDR; however, it explores only a local-optimum given that Genetic Algorithms do not fully explore the optimal distribution.
- **PDR-GA:** Combines previous two datasets, which entails a larger coverage of both local-optimum examples, from GA, and more suboptimal examples from PDR.
- **Random:** Has the largest coverage; however, it contains suboptimal examples generated by a random heuristic.

The results in Table 1 show that CDQAC trained in Random (Greedy:  $10.68\% \pm 0.51\%$ , Sampling:  $5.86\% \pm 0.3\%$ ) or PDR-GA (Greedy:  $11.31\% \pm 1.33\%$ , Sampling:  $5.87\% \pm 0.51\%$ ), compared to trained in PDR (Greedy:  $12.34\% \pm 1.72\%$ , Sampling:  $6.57\% \pm 0.76\%$ ) or GA (Greedy:  $13.06\% \pm 2.11\%$ , Sampling:  $6.43\% \pm 0.87\%$ ). The reason why CDQAC achieved higher performance with PDR-GA or Random is through **unconfirmed pessimism** Jin et al. (2021). Jin et al. (2021) states that the expert dataset, like GA, contains only a narrow perspective of the possible state action spaces. Therefore, CDQAC, a pessimistic offline RL approach, can never confirm the artificial pessimism. Moreover, since the GA dataset is focused around a specific local optimum, since GA is not an optimal approach, CDQAC is also enforced around this optimum. By including more suboptimal examples, CDQAC is able to confirm the artificial pessimism. This is supported by Kumar et al. (2022), who state that noisy-expert datasets enable counterfactual examples for the offline RL method, to show which action it should not take.

Additionally, in FJSP and JSP the reward  $r_t = \max_{O_{i,j} \in \mathcal{O}} C(O_{i,j}, s_t) - \max_{O_{i,j} \in \mathcal{O}} C(O_{i,j}, s_{t+1})$ , and therefore the cumulative reward  $C_{\max} = \max_{O_{i,j} \in \mathcal{O}} C(O_{i,j}, s_0) + \sum_{t=0}^H r_t$ , is equal to the makespan including the partial makespan  $\max_{O_{i,j} \in \mathcal{O}} C(O_{i,j}, s_0)$  in the time step  $t = 0$ . Table 3 shows that this enables CDQAC to learn the exact value of bad and good action, rather than proxy rewards as are used in most other offline benchmarks, such as robotics where the only objective is to perform a task successfully Fu et al. (2020). This results in a sparse reward environment for these problems, where the reward is only +1, if the task was completed successfully, whereby the horizon is dependent on the policy. This differs from our scheduling setting, with a dense reward, which directly aligns with the objective, aka the makespan, and where both a random and expert heuristic will generate data with the same trajectory length.

## F.2 Intrinsic Uncertainty and Trajectory Stitching

Our prior section explained why CDQAC achieved a higher performance with PDR-GA than with the GA dataset. Yet, this does not fully explain why CDQAC achieves the highest performance with Random. To fully explain these results, we require additional explanation on both **Intrinsic Uncertainty** and **Trajectory Stitching**. Jin et al. (2021) states that Intrinsic Uncertainty as the uncertainty the Offline RL method is due to the lack of state-action pairs taken by the optimal policy  $\pi^*$  found in the training dataset. Since all our training datasets are suboptimal, we cannot guarantee that they will contain all the state action pairs taken by  $\pi^*$ ; however, a greater coverage in the dataset significantly reduces this uncertainty.

Moreover, we know that finding an optimal trajectory  $\tau^*$  is infeasible in a dataset, given that it is equal to  $P(\tau^*) \propto |\mathcal{A}|^{-H}$ , where  $H$  is the length of the episode. In comparison, finding a state-action pair  $(s, a)^*$  taken by the optimal policy  $\pi^*$  is linearly proportional to the size of the action space  $P((s, a)^*) \propto |\mathcal{A}|^{-1}$ . This property enables CDQAC to see the random dataset as a “bag of puzzle pieces”, from which it must learn a new policy through **trajectory stitching**. Moreover, since all datasets are generated with suboptimal heuristics, we can state that none of these “bags of puzzle pieces” are complete; however, greater coverage, as shown in Table 2, increases the likelihood of having all the puzzle pieces, aka, reducing the intrinsic uncertainty.

A lower intrinsic uncertainty improves the ability of CDQAC to for **trajectory stitching**. In the following, we illustrate this through two trajectories,  $\tau_1$  who starts well but ends at a negative state, and  $\tau_2$ , who perform poorly at the start but performs well at the start:

$$\begin{aligned} \tau_1 &= s_{\text{good start}} \rightarrow \dots \rightarrow s_{\text{intersect}} \rightarrow \dots \rightarrow s_{\text{bad end}} && \text{(Starts well, ends poorly).} \\ \tau_2 &= s_{\text{bad start}} \rightarrow \dots \rightarrow s_{\text{intersect}} \rightarrow \dots \rightarrow s_{\text{good end}} && \text{(Starts poorly, ends good).} \end{aligned}$$

This results in both  $\tau_1$  and  $\tau_2$  having low returns. However, they share a common state (or a state with a similar embedding)  $s_{\text{intersect}}$ . During Q-learning, the high value of the end of  $\tau_2$  propagates backward to  $s_{\text{intersect}}$ .

$$Q(s_{\text{intersect}}, a_{\tau_2}) \leftarrow r + \gamma \max Q(s_{\text{next}}, \cdot)$$

Subsequently, when updating the value of the start of  $\tau_1$ :

$$Q(s_{\text{prev}}, a_{\tau_1}) \leftarrow r + \gamma Q(s_{\text{intersect}}, \cdot)$$

The agent learns a composite policy  $\pi_{\text{new}} : s_{\text{start}} \rightarrow \dots \rightarrow s_{\text{intersect}} \rightarrow \dots \rightarrow s_{\text{good}}$ . This new trajectory  $\tau_{\text{new}}$  never existed in the dataset, yet it is constructed from valid transitions found within it.

**Limitations of GA Data:** In contrast, the GA dataset often lacks the diversity required for this stitching. Because the heuristic focuses on a narrow region of the state space, trajectories tend to converge to a local optimum  $s_{\text{local}}$  rather than the global optimum  $s_{\text{global}}$ . Consider two GA trajectories:

$$\begin{aligned} \tau_{\text{GA}_1} &= s_{\text{start}} \rightarrow \dots \rightarrow s_{\text{intersect}} \rightarrow s_{\text{path}_A} \rightarrow \dots \rightarrow s_{\text{local}} \\ \tau_{\text{GA}_2} &= s_{\text{start}} \rightarrow \dots \rightarrow s_{\text{intersect}} \rightarrow s_{\text{path}_{A'}} \rightarrow \dots \rightarrow s_{\text{local}} \end{aligned}$$

Even if a better path exists via a different action at  $s_{\text{decision}}$  (e.g.,  $s_{\text{decision}} \rightarrow s_{\text{path}_B} \rightarrow s_{\text{global}}$ ), the GA dataset never explores the transition  $(s_{\text{decision}}, s_{\text{path}_B})$ . Consequently, these “puzzle pieces” are missing, and

Table 9: Ablation on state-action coverage via  $\varepsilon$ -PDR. CDQAC trained on Random vs.  $\varepsilon$ -greedy mixtures of PDR/PDR-GA with  $\varepsilon \in \{0.1, 0.2\}$ , averaged over training sizes  $\{10 \times 5, 15 \times 10, 20 \times 10\}$ . SACo: state-action coverage relative to PDR ( $\uparrow$ ); Gap (%): mean  $\pm$  std over four seeds ( $\downarrow$ );  $p$ -values from a Wilcoxon signed-rank test against Random on per-instance gaps. **Bold** marks the best per column (SACo, Gap) and  $p < 0.05$  ( $p$ -value column).

Dataset	Greedy Gap (%) $\downarrow$	Sampling Gap (%) $\downarrow$	SACo $\uparrow$	$p$ -value vs. Random
Random	10.68 $\pm$ 0.51	5.87 $\pm$ 0.30	7.70 $\pm$ 0.77	—
$\varepsilon$ -PDR ( $\varepsilon = 0.1$ )	10.97 $\pm$ 0.42	5.98 $\pm$ 0.23	6.82 $\pm$ 0.56	<b>0.006</b>
$\varepsilon$ -PDR-GA ( $\varepsilon = 0.1$ )	10.76 $\pm$ 0.47	5.86 $\pm$ 0.32	7.83 $\pm$ 0.68	0.364
$\varepsilon$ -PDR ( $\varepsilon = 0.2$ )	10.87 $\pm$ 0.53	5.93 $\pm$ 0.62	7.37 $\pm$ 0.61	0.054
$\varepsilon$ -PDR-GA ( $\varepsilon = 0.2$ )	<b>10.45 <math>\pm</math> 0.36</b>	<b>5.81 <math>\pm</math> 0.32</b>	<b>8.38 <math>\pm</math> 0.72</b>	0.209

CDQAC, constrained by pessimism, is forced to replicate the local optimum rather than stitching a path to the global optimum. This explanation also relates to our results about dataset properties in Table 3. Namely, if either the reward is sparse or an proxy, CDQAC is unable to learn policy with Random, that will outperform compared to when trained on GA or PDR-GA. With a sparse reward, CDQAC is unable to differentiate which path to take based on  $\tau_1$  or  $\tau_2$ , since both will have approximately the same makespan. This issue also arises with the proxy reward, since we cannot see the difference between a bad state-action pair and an even worse state-action pair.

Moreover, our results in Table 3 show that if neither the expert dataset, GA, nor the Random dataset has access to terminal states, which means it only has access until  $s_{\text{intersect}}$ , CDQAC cannot learn a well performing policy. This means that if we do not have access to the end states, CDQAC and offline RL in general cannot learn a good policy, as shown in prior work Kumar et al. (2020); Fu et al. (2020) with the Atari Benchmark, where a random policy would result in a short episode, aka a Game Over, not containing information about the end states. This connects back to the earlier definition of **Intrinsic Uncertainty** Jin et al. (2021), since the uncertainty will be maximal given that we do not have access to any of the end states.

### F.3 Empirical Validation

The analyses in App. F.1–F.2 attribute CDQAC’s strong performance on the Random dataset to higher state–action coverage (SACo) and the resulting reduction in intrinsic uncertainty. An alternative explanation, however, is that CDQAC has some implicit affinity for purely random data — for instance, that its conservative regularization or quantile critic happens to cooperate well with uniform action sampling specifically. To distinguish coverage from data source, we construct  $\varepsilon$ -PDR datasets in which, at every timestep, the behavioural policy selects a random feasible action with probability  $\varepsilon$  and a PDR action with probability  $1 - \varepsilon$ . We test  $\varepsilon \in \{0.1, 0.2\}$  in two configurations: (i)  $\varepsilon$ -PDR uses 100 trajectories per instance, drawing a fresh PDR rule at the start of every trajectory; (ii)  $\varepsilon$ -PDR-GA additionally retains the 200 GA trajectories per instance and adds 50  $\varepsilon$ -PDR trajectories. Both configurations preserve heuristic structure within trajectories, in contrast to Random. We compare each setup against CDQAC trained on Random using a Wilcoxon signed-rank test on per-instance gaps ( $N=36$ ); all training is on  $10 \times 5$  instances.

The pattern in Table 9 aligns with the coverage hypothesis rather than the synergy hypothesis. The only configuration that significantly underperforms Random is  $\varepsilon$ -PDR with  $\varepsilon = 0.1$  ( $p \approx 0.006$ ), which is also the only configuration whose SACo falls below Random’s (6.82 vs. 7.70). The remaining three configurations are statistically indistinguishable from Random ( $p \in \{0.054, 0.209, 0.364\}$ ), and  $\varepsilon$ -PDR-GA ( $\varepsilon = 0.2$ ) attains both the highest SACo (8.38) and the lowest gap on both evaluation modes despite retaining the full GA component of PDR-GA. If CDQAC required random data *per se*, GA-derived structure should have hurt performance.

## G Hyperparameters

In Table 10, we state the hyperparameters used in all our experiments. Furthermore, we used two layers of the DAN network, whereby we concatenated the output of each head for the first layer and averaged the

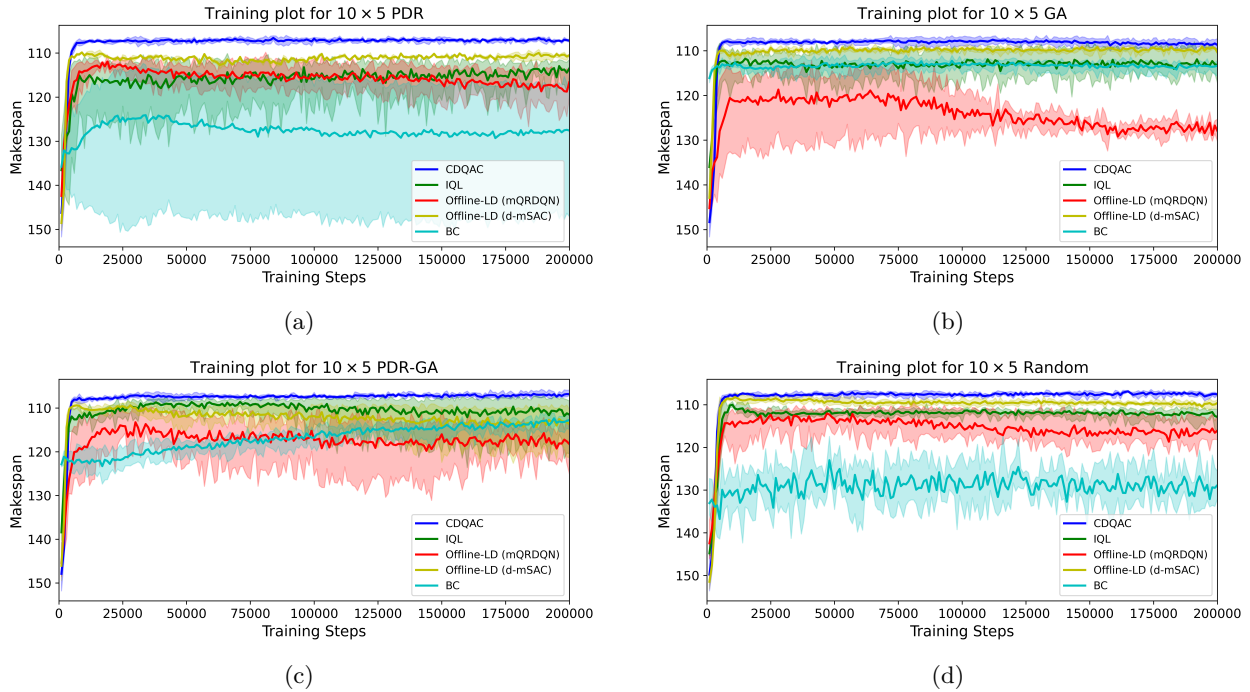


Figure 5: The training plots when trained of FJSP instances of size  $10 \times 5$  for BC, CDQAC, IQL and Offline-LD, both mQRDQN and d-mSAC. Fig. 5a shows the training plots when trained on the PDR dataset, Fig. 5b with the GA dataset, Fig. 5c with the PDR-GA dataset, and Fig. 5d the Random dataset. The line is average makespan over four different seeds and the shaded area is minimal and maximal makespan of these seeds. We evaluate each method at every 1,000 steps of offline training.

heads for the second layer. Both the value stream  $V_\theta$  and the advantage stream  $A_\theta$ , consist of three layers, each having 64 neurons. For each seed, we train for 200,000 steps, with a batch size of 256. We normalize all features in the training dataset. We used ADAM (Kingma, 2014) optimizer.

Table 10: Hyperparameter settings CDQAC.

Hyperparameter	Value
Policy Frequency Update $\eta$	4
CQL Strength $\alpha_{CQL}$	0.05
Number of quantile fractions $N$	64
Learning rate quantile critic $\ell_\theta$	$2 \times 10^{-4}$
Learning rate policy $\ell_\psi$	$2 \times 10^{-5}$
Target Update Frequency $\rho$	0.005
Entropy Coefficient $\lambda$	0.005
Batch Size	256
Training Steps	200,000
Network Parameters	
Layers DAN network	2
Output Dimension DAN	(32, 8)
Number of Heads $H$	4
Hidden Dimension Quantile Critic $Z_\theta$	64
Hidden Layers Quantile Critic $Z_\theta$	2
Hidden Dimension Policy $\pi_\psi$	64
Hidden Layers Policy $\pi_\psi$	2

## H Training Plots

Fig. 5, Fig. 6, and Fig 7 show the training plots for all the methods used in our FJSP evaluations (Tables 1, 12, 13, and 14). In the figure, we note that CDQAC converges in significantly fewer steps than the 200,000

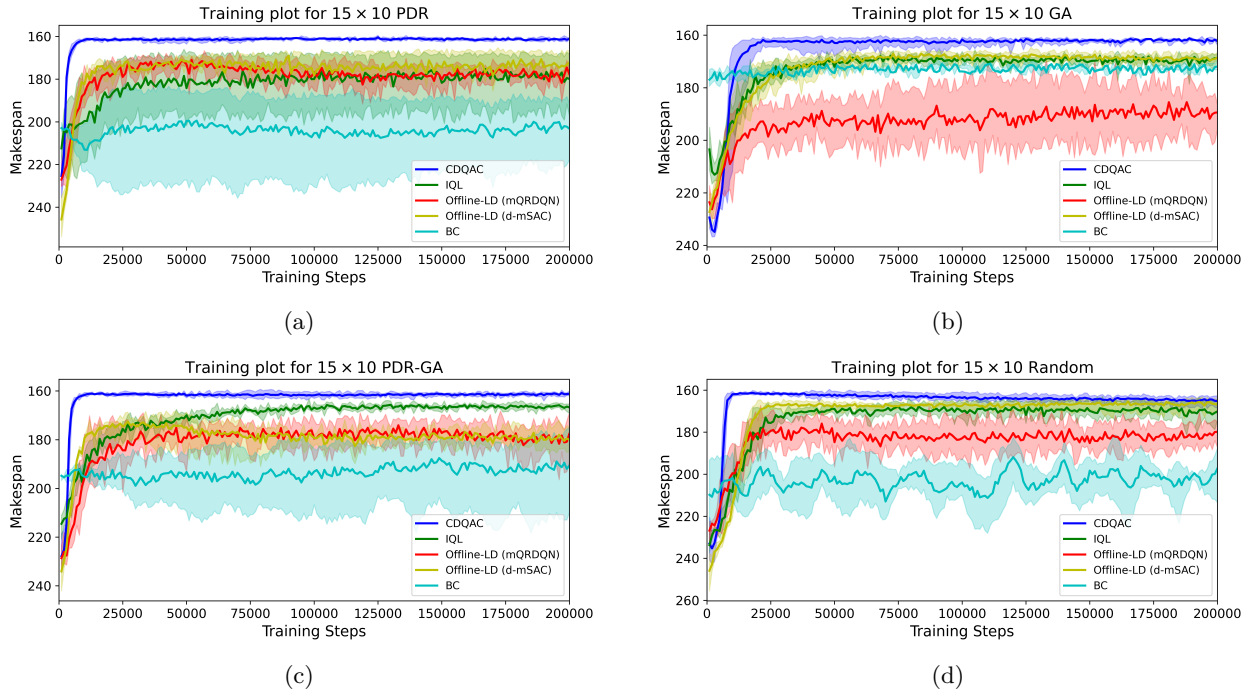


Figure 6: The training plots when trained of FJSP instances of size  $15 \times 10$  for BC, CDQAC, IQL and Offline-LD, both mQRDQN and d-mSAC. Fig. 6a shows the training plots when trained on the PDR dataset, Fig. 6b with the GA dataset, Fig. 6c with the PDR-GA dataset, and Fig. 6d the Random dataset. The line is average makespan over four different seeds and the shaded area is minimal and maximal makespan of these seeds. We evaluate each method at every 1,000 steps of offline training.

training steps used. For example, for  $10 \times 5$  CDQAC requires around 10,000 steps according to Fig. 5, and around 25,000 training steps for  $15 \times 10$  as seen in Fig. 6.

Based on the training plots, we can determine that CDQAC achieves the most stable training with the highest average Makespan in each evaluation step. The only exception is with the Random dataset for  $20 \times 10$  (Fig. 7d), where both Offline-LD (d-mSAC) and IQL are more stable and have a higher evaluation at the last training step. However, CDQAC for all other datasets and training datasets. For example, Offline-LD (d-mSAC) cannot learn a policy with the PDR-GA dataset for  $20 \times 10$  (Fig. 7c), and IQL with the PDR dataset for all training sizes. (Figs 5a, 6a and 7a). Lastly, we can notice for all training plots that CDQAC converges significantly faster than the other offline RL methods.

## I Additional Results

### I.1 Ablation Study

Sect. 5.5 reports the ablation of CDQAC’s four components — *CQL regularization*, *Quantile Critic*, *Dueling Architecture*, and *Delayed Policy Update* — averaged across instance sizes and evaluation modes. Table 11 expands the bars of Fig. 3 along two additional dimensions: the three training sizes ( $10 \times 5$ ,  $15 \times 10$ ,  $20 \times 10$ ), and a separation between *in-distribution* performance (generated instances matching the training distribution) and *out-of-distribution* performance (the Brandimarte (Brandimarte, 1993) and Hurink (Hurink et al., 1994) benchmarks).

Two patterns emerge. First, removing the Quantile Critic or the CQL regularization increases the gap by only 1–5%, with little variation across instance sizes, consistent with the averages reported in Section 5.5. Second, the contributions of the Delayed Policy Update and the Dueling Architecture grow sharply with instance size: removing the Delayed Policy Update increases the gap by 11.7% at  $10 \times 5$ , 38.5% at  $15 \times 10$ , and 112.8% at

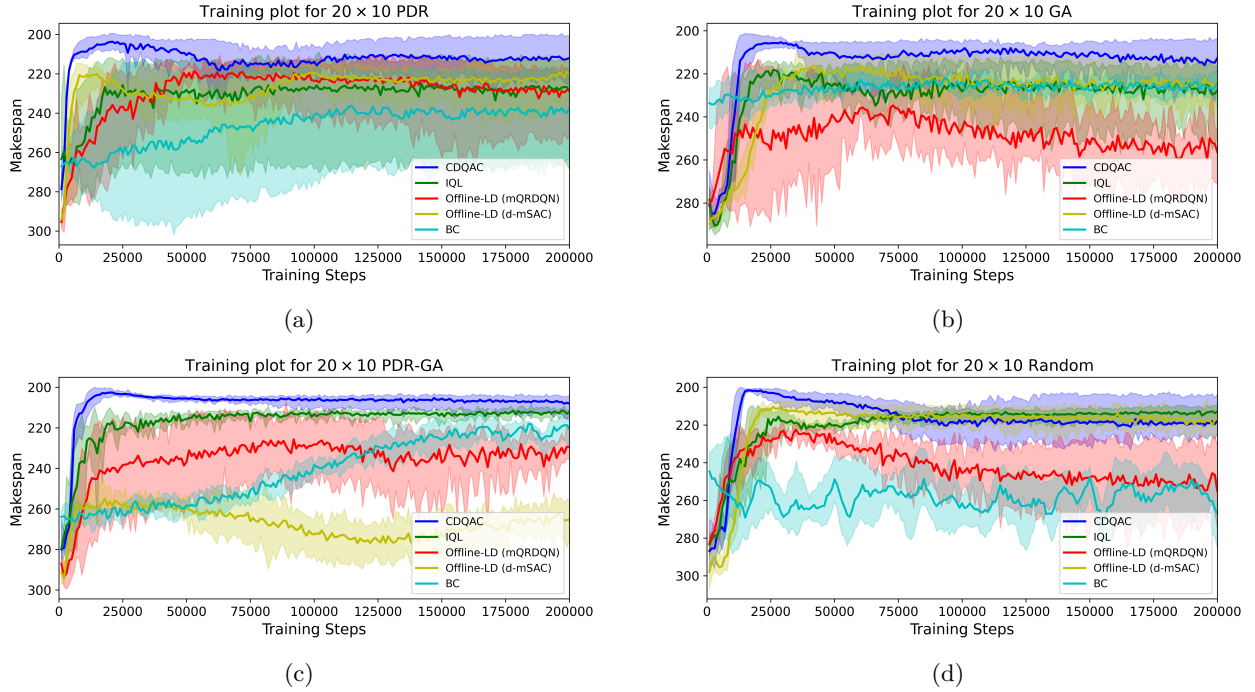


Figure 7: The training plots when trained of FJSP instances of size  $20 \times 10$  for BC, CDQAC, IQL and Offline-LD, both mQRDQN and d-mSAC. Fig. 7a shows the training plots when trained on the PDR dataset, Fig. 7b with the GA dataset, Fig. 7c with the PDR-GA dataset, and Fig. 7d the Random dataset. The line is average makespan over four different seeds and the shaded area is minimal and maximal makespan of these seeds. We evaluate each method at every 1,000 steps of offline training.

Table 11: Ablation study of CDQAC components, trained on the Random dataset across instance sizes. We report the mean and standard deviation of the gap (%) across four seeds on In-Distribution instances (generated and similar to training set) and Out-of-Distribution FJSP benchmarks (Brandimarte and Hurink). Avg  $\Delta$ (%) is the mean percentage difference in gap relative to Standard (full CDQAC). **Bold** indicates best result per column within each instance size.

Component	In-Distribution (Gap %)		Out-of-Distribution (Gap %)		Avg $\Delta$ (%)	
	Greedy	Sampling	Greedy	Sampling		
$10 \times 5$	Standard	<b>11.19 <math>\pm</math> 0.35</b>	<b>5.87 <math>\pm</math> 0.14</b>	<b>10.45 <math>\pm</math> 0.39</b>	<b>6.05 <math>\pm</math> 0.10</b>	0.00%
	w/o CQL	11.49 $\pm$ 0.26	5.98 $\pm$ 0.17	10.59 $\pm$ 0.27	6.16 $\pm$ 0.14	+1.93%
	w/o Quantile	11.72 $\pm$ 0.53	6.05 $\pm$ 0.11	10.50 $\pm$ 0.21	6.24 $\pm$ 0.14	+2.86%
	w/o Dueling	11.59 $\pm$ 0.53	5.99 $\pm$ 0.27	10.97 $\pm$ 0.43	6.45 $\pm$ 0.30	+4.30%
	w/o Delayed Policy	12.27 $\pm$ 0.49	6.30 $\pm$ 0.14	12.46 $\pm$ 1.12	6.69 $\pm$ 0.27	+11.70%
$15 \times 10$	Standard	<b>12.04 <math>\pm</math> 0.59</b>	6.70 $\pm$ 0.62	<b>9.81 <math>\pm</math> 0.36</b>	<b>5.10 <math>\pm</math> 0.26</b>	0.00%
	w/o CQL	12.42 $\pm$ 0.36	7.03 $\pm$ 0.76	10.56 $\pm$ 0.81	5.25 $\pm$ 0.26	+4.88%
	w/o Quantile	12.35 $\pm$ 0.28	<b>6.57 <math>\pm</math> 0.07</b>	10.81 $\pm$ 0.26	5.47 $\pm$ 0.08	+5.13%
	w/o Dueling	12.42 $\pm$ 0.31	6.65 $\pm$ 0.28	10.94 $\pm$ 0.47	5.41 $\pm$ 0.19	+5.77%
	w/o Delayed Policy	13.13 $\pm$ 0.43	7.03 $\pm$ 0.30	17.24 $\pm$ 1.96	7.87 $\pm$ 1.04	+38.46%
$20 \times 10$	Standard	<b>5.20 <math>\pm</math> 0.66</b>	2.87 $\pm$ 0.73	<b>10.90 <math>\pm</math> 0.73</b>	5.50 $\pm$ 0.22	0.00%
	w/o CQL	5.32 $\pm$ 0.70	2.78 $\pm$ 0.72	11.18 $\pm$ 1.00	<b>5.39 <math>\pm</math> 0.26</b>	+0.74%
	w/o Quantile	5.44 $\pm$ 0.30	3.11 $\pm$ 0.22	11.28 $\pm$ 0.54	5.83 $\pm$ 0.19	+4.68%
	w/o Dueling	6.60 $\pm$ 1.65	<b>2.66 <math>\pm</math> 0.34</b>	13.30 $\pm$ 1.99	6.02 $\pm$ 0.54	+16.73%
	w/o Delayed Policy	15.37 $\pm$ 5.51	6.02 $\pm$ 2.00	21.31 $\pm$ 4.61	10.41 $\pm$ 2.47	+112.75%

$20 \times 10$ , with the Dueling Architecture following the same trend on a smaller scale (4.3%  $\rightarrow$  5.8%  $\rightarrow$  16.7%). The in-distribution / out-of-distribution split further shows that this scaling does not reflect overfitting: at  $20 \times 10$ , removing the Delayed Policy Update degrades both the in-distribution gap (from 5.20% to 15.37%) and the out-of-distribution gap (from 10.90% to 21.31%), indicating that without delayed updates CDQAC

fails to train rather than fails to generalize. These findings reinforce the action-space argument in Section 5.5: the Delayed Policy Update and the Dueling Architecture are the two components responsible for keeping CDQAC stable as the combinatorial action space grows.

## I.2 Results Offline RL

Table 12: Results of FJSP offline RL comparison  $10 \times 5$ , for all training datasets (PDR, GA, PDR-GA, and Random). The columns show the evaluation benchmarks sets and the rows the methods. The mean and standard deviation of the gap (%) are reported from four different seeds. **Bold** indicates best result (lowest gap) for either the Greedy and Sampling (100 solutions) evaluation, for a given training dataset.

	Generated $10 \times 5$		Brandimarte (mk)		Hurink edata		Hurink rdata		Hurink vdata	
	Greedy	Sampling	Greedy	Sampling	Greedy	Sampling	Greedy	Sampling	Greedy	Sampling
PDR										
BC	31.79±1.96	10.45±0.84	72.5±5.36	33.7±1.42	31.03±2.02	13.93±0.89	30.04±3.18	12.58±1.12	14.97±2.2	4.16±0.84
Offline-LD (mQRDQN)	15.4±1.2	14.39±0.12	22.81±3.76	25.07±0.27	25.54±2.4	12.38±0.06	18.74±2.55	10.24±0.09	11.77±1.11	3.37±0.05
Offline-LD (d-mSAC)	15.26±0.85	8.16±0.11	43.74±5.43	23.18±3.39	22.17±2.1	10.18±0.8	21.93±3.5	9.34±2.36	7.55±0.76	1.3±0.2
IQL	15.58±0.47	8.13±0.17	41.75±3.87	21.89±1.15	22.87±1.84	11.25±0.98	21.36±3.56	8.12±1.0	7.32±0.89	1.43±0.4
<b>CDQAC</b>	<b>11.49±0.38</b>	<b>5.64±0.08</b>	<b>12.43±1.45</b>	<b>8.3±0.14</b>	<b>15.11±1.06</b>	<b>9.68±0.57</b>	<b>10.81±0.22</b>	<b>5.54±0.12</b>	<b>3.69±0.25</b>	<b>0.78±0.02</b>
GA										
BC	14.63±0.7	8.91±0.3	16.03±1.81	15.03±0.46	15.27±0.58	8.79±0.31	11.36±0.59	6.69±0.17	4.48±0.23	1.43±0.07
Offline-LD (mQRDQN)	17.28±3.88	14.52±0.08	33.45±8.26	26.62±0.62	29.64±3.0	12.55±0.07	22.84±1.78	10.47±0.2	14.13±1.99	3.51±0.06
Offline-LD (d-mSAC)	<b>11.38±0.64</b>	<b>5.29±0.1</b>	23.47±3.33	12.05±1.37	21.55±3.24	<b>9.23±1.23</b>	16.32±2.16	5.99±0.47	11.37±1.92	2.89±1.02
IQL	13.02±0.86	7.32±0.18	26.71±1.01	14.57±0.71	25.67±2.1	10.72±0.58	17.37±1.81	6.75±0.31	10.69±1.76	2.14±0.62
<b>CDQAC</b>	11.62±0.35	6.09±0.22	<b>15.51±1.0</b>	<b>9.58±0.76</b>	<b>14.87±0.25</b>	9.45±0.54	<b>10.44±0.4</b>	<b>5.39±0.2</b>	<b>3.24±0.3</b>	<b>0.65±0.01</b>
PDR-GA										
BC	16.79±1.13	8.86±0.08	57.26±4.68	27.0±1.77	24.37±1.15	11.17±0.3	28.38±1.03	10.99±1.29	15.72±1.5	3.18±1.17
Offline-LD (mQRDQN)	14.7±0.99	14.33±0.04	21.77±1.22	25.27±0.45	25.53±2.79	12.25±0.11	19.34±2.61	10.33±0.06	11.94±2.17	3.45±0.05
Offline-LD (d-mSAC)	12.1±0.65	5.9±0.48	19.49±2.67	11.17±0.68	19.04±1.61	<b>8.82±0.61</b>	13.27±0.84	5.58±0.31	7.59±1.86	1.32±0.32
IQL	12.4±0.24	7.22±0.09	33.13±4.61	19.43±2.27	26.44±3.42	11.69±1.26	21.42±3.37	7.83±0.67	11.24±1.83	2.06±0.3
<b>CDQAC</b>	<b>11.16±0.43</b>	<b>5.88±0.37</b>	<b>14.24±1.23</b>	<b>8.79±0.74</b>	<b>15.3±0.57</b>	9.84±0.38	<b>10.96±0.56</b>	<b>5.51±0.16</b>	<b>3.59±0.31</b>	<b>0.72±0.03</b>
Random										
BC	25.59±2.86	14.91±0.05	31.74±2.78	26.95±0.2	22.12±1.46	12.26±0.06	17.16±2.35	10.48±0.17	7.98±2.22	3.46±0.08
Offline-LD (mQRDQN)	14.41±0.87	14.17±0.14	21.42±1.44	25.0±1.03	19.05±1.5	11.93±0.11	14.85±1.64	9.98±0.15	7.91±1.68	3.22±0.15
Offline-LD (d-mSAC)	13.29±0.45	6.26±0.27	16.62±0.6	9.49±0.37	16.12±1.43	<b>8.24±0.32</b>	12.13±0.99	5.67±0.23	4.14±0.74	0.87±0.08
IQL	15.64±1.2	8.98±0.15	33.11±5.9	18.73±1.42	26.91±4.2	11.5±1.29	17.65±3.0	7.5±0.87	12.85±5.68	2.75±1.45
<b>CDQAC</b>	<b>11.19±0.35</b>	<b>5.87±0.14</b>	<b>13.78±0.78</b>	<b>8.67±0.21</b>	<b>14.53±0.41</b>	9.54±0.39	<b>10.4±0.36</b>	<b>5.3±0.22</b>	<b>3.1±0.22</b>	<b>0.68±0.03</b>

In this section, we provide a comprehensive overview of the results discussed in Sect.5.1 and Table1, where we compare our proposed method, CDQAC, to Offline-LD (Remmerden et al., 2025). Table 1 presents the average performance across all evaluation instance sets—both generated and benchmark—for each training size ( $10 \times 5$ ,  $15 \times 10$ , and  $20 \times 10$ ). The detailed results for each evaluation set are reported in Table 12 (training size  $10 \times 5$ ), Table 13 ( $15 \times 10$ ), and Table 14 ( $20 \times 10$ ).

As shown in Tables 12, 13, and 14, CDQAC consistently outperforms both versions of Offline-LD in nearly all evaluations. There are only a few exceptions: in Table 12, Offline-LD (d-mSAC) marginally exceeds CDQAC in the generated instances and Hurink edata using the sampling evaluation when trained on the GA dataset, as well as on Hurink edata with the sampling evaluation when both methods are trained on the Random dataset. Nevertheless, CDQAC shows better performance on the remaining evaluation sets for both the GA and Random training sets. Furthermore, with larger training sizes,  $15 \times 10$  (Table 13) and  $20 \times 10$  (Table 14), CDQAC consistently outperforms Offline-LD, and the performance margins widen as the instance size increases. These findings indicate that CDQAC scales more efficiently to larger instance sizes, and is generally an improvement over the offline RL baseline, Offline-LD.

Analyzing CDQAC’s performance across different instance sizes and training datasets, we observe that for both  $10 \times 5$  (Table 12) and  $15 \times 10$  (Table 13), CDQAC achieves the worst performance when trained on the GA dataset across all evaluation sets. In contrast, for  $20 \times 10$ , CDQAC trained on the GA dataset achieves the best performance on generated instances (Greedy:  $5.01\% \pm 0.28\%$ ), while training on PDR yields the worst results (Greedy:  $9.38\% \pm 6.1\%$ ), accompanied by a high standard deviation. This higher standard deviation with PDR suggests instability during training, as one of the four runs did not train effectively. Additionally, we find that, when trained on GA, CDQAC struggles to generalize to unseen evaluation instances compared to when trained on more diverse datasets, such as Random and PDR-GA. This further supports the conclusion

Table 13: Results of FJSP offline RL comparison  $15 \times 10$ , for all training datasets (PDR, GA, PDR-GA, and Random). The columns show the evaluation benchmarks sets and the rows the methods. The mean and standard deviation of the gap (%) are reported from four different seeds. **Bold** indicates best result (lowest gap) for either the Greedy and Sampling (100 solutions) evaluation, for a given training dataset.

	Generated $15 \times 10$		Brandimarte (mk)		Hurink edata		Hurink rdata		Hurink vdata	
	Greedy	Sampling	Greedy	Sampling	Greedy	Sampling	Greedy	Sampling	Greedy	Sampling
PDR										
BC	36.31±3.88	13.58±0.88	58.23±13.17	30.47±3.21	28.66±6.05	12.13±1.48	23.93±2.1	8.57±1.13	10.38±1.4	2.04±0.49
Offline-LD (mQRDQN)	17.36±1.17	20.28±0.09	22.89±1.89	24.88±0.22	30.32±1.54	12.51±0.17	19.93±1.61	10.2±0.15	10.01±2.47	3.33±0.07
Offline-LD (d-mSAC)	16.37±0.5	10.54±0.16	39.55±6.46	23.6±2.54	23.63±6.52	11.85±3.21	14.93±2.03	6.43±0.16	5.82±0.95	1.26±0.22
IQL	16.35±0.53	10.51±0.22	30.95±2.93	19.75±0.77	20.5±0.29	<b>9.98±0.19</b>	14.08±1.57	6.38±0.08	5.54±0.34	1.04±0.06
<b>CDQAC</b>	<b>12.21±0.37</b>	<b>6.48±0.15</b>	<b>14.6±0.78</b>	<b>9.6±0.1</b>	<b>17.67±1.49</b>	<b>10.77±0.35</b>	<b>11.67±0.6</b>	<b>5.76±0.08</b>	<b>3.94±0.43</b>	<b>0.87±0.16</b>
GA										
BC	17.21±0.31	13.41±0.09	28.88±1.67	18.13±0.21	17.73±0.78	9.22±0.12	14.43±0.44	6.97±0.05	11.03±0.61	1.8±0.07
Offline-LD (mQRDQN)	24.67±2.98	20.47±0.07	45.24±4.87	27.03±0.38	34.83±1.61	12.9±0.11	28.1±1.6	10.72±0.07	19.63±1.82	3.78±0.06
Offline-LD (d-mSAC)	16.11±0.71	8.74±0.1	29.23±1.9	14.89±0.51	31.93±2.12	13.69±0.86	22.88±1.09	8.39±0.13	16.12±2.14	4.71±0.56
IQL	15.54±0.84	11.08±0.2	26.69±3.37	16.28±0.79	26.84±3.04	<b>11.78±0.84</b>	20.41±2.1	7.72±0.47	14.15±2.5	3.26±1.04
<b>CDQAC</b>	<b>12.3±0.45</b>	<b>6.19±0.24</b>	<b>19.6±4.61</b>	<b>10.22±1.76</b>	<b>23.53±6.23</b>	<b>11.8±2.82</b>	<b>14.37±3.46</b>	<b>6.13±0.83</b>	<b>7.46±3.69</b>	<b>1.63±0.96</b>
PDR-GA										
BC	23.94±4.08	13.35±0.76	57.21±3.94	26.53±0.61	27.61±1.06	11.73±0.68	22.35±2.5	8.37±0.79	12.78±1.97	2.67±0.31
Offline-LD (mQRDQN)	18.15±1.12	20.34±0.04	23.98±3.91	25.53±0.44	27.62±2.08	12.52±0.23	21.92±1.47	10.42±0.14	12.19±2.4	3.5±0.1
Offline-LD (d-mSAC)	17.42±0.65	9.36±0.36	35.9±4.16	17.54±1.75	34.09±3.15	14.81±1.36	21.91±1.3	8.75±0.29	14.99±1.35	4.62±0.22
IQL	15.33±0.52	10.5±0.13	28.15±1.59	19.1±0.66	25.06±2.39	11.43±0.43	16.4±2.51	6.69±0.24	6.56±2.62	1.22±0.26
<b>CDQAC</b>	<b>12.28±0.26</b>	<b>6.15±0.47</b>	<b>14.75±1.53</b>	<b>8.72±0.59</b>	<b>18.02±4.44</b>	<b>9.55±1.42</b>	<b>11.44±0.88</b>	<b>5.44±0.28</b>	<b>3.51±0.91</b>	<b>0.78±0.15</b>
Random										
BC	30.41±3.73	20.87±0.09	36.61±4.36	26.66±0.33	25.66±2.26	12.33±0.09	22.56±3.89	10.5±0.12	10.92±3.12	3.58±0.03
Offline-LD (mQRDQN)	16.95±0.54	20.21±0.07	29.14±4.62	25.6±0.39	29.07±3.02	12.58±0.24	20.17±2.17	10.24±0.12	12.83±1.86	3.41±0.07
Offline-LD (d-mSAC)	15.02±0.43	8.17±0.31	20.44±1.58	11.27±0.49	30.92±3.15	14.52±1.5	18.06±1.22	7.46±0.33	9.97±1.41	2.32±0.45
IQL	15.58±1.64	13.75±0.28	24.5±2.93	18.12±0.42	24.63±4.43	11.54±0.78	19.69±2.85	8.81±0.53	12.43±3.42	3.75±0.86
<b>CDQAC</b>	<b>12.04±0.59</b>	<b>6.7±0.62</b>	<b>13.58±0.66</b>	<b>8.73±0.73</b>	<b>14.56±0.55</b>	<b>8.51±0.52</b>	10.77±0.36	<b>5.22±0.12</b>	<b>3.16±0.1</b>	<b>0.67±0.02</b>

Table 14: Results of FJSP offline RL comparison  $20 \times 10$ , for all training datasets (PDR, GA, PDR-GA, and Random). The columns show the evaluation benchmarks sets and the rows the methods. The mean and standard deviation of the gap (%) are reported from four different seeds. **Bold** indicates best result (lowest gap) for either the Greedy and Sampling (100 solutions) evaluation, for a given training dataset.

	Generated $20 \times 10$		Brandimarte (mk)		Hurink edata		Hurink rdata		Hurink vdata	
	Greedy	Sampling	Greedy	Sampling	Greedy	Sampling	Greedy	Sampling	Greedy	Sampling
PDR										
BC	33.37±2.71	9.29±0.78	65.13±6.3	34.94±2.12	27.47±3.56	12.91±2.18	24.42±4.57	9.11±0.96	9.03±2.55	1.27±0.04
Offline-LD (mQRDQN)	27.6±5.91	14.82±0.12	33.83±2.4	26.54±1.25	31.03±2.1	12.61±0.23	28.02±3.74	10.55±0.11	18.73±2.71	3.56±0.09
Offline-LD (d-mSAC)	15.43±3.82	8.38±1.08	55.97±4.05	33.3±1.67	33.17±4.2	15.66±2.26	23.86±1.87	8.91±0.87	9.9±2.93	2.11±0.9
IQL	10.43±1.11	6.77±0.33	45.31±3.96	24.95±1.83	25.31±4.42	11.8±1.26	16.59±1.26	7.19±0.26	<b>5.06±0.41</b>	<b>1.06±0.06</b>
<b>CDQAC</b>	<b>9.38±6.1</b>	<b>4.38±3.47</b>	<b>16.65±0.5</b>	<b>9.7±0.7</b>	<b>21.5±5.18</b>	<b>11.23±1.97</b>	<b>15.53±3.05</b>	<b>6.98±1.29</b>	8.47±4.0	2.94±2.31
GA										
BC	11.73±0.59	8.4±0.09	24.69±1.69	18.36±0.44	17.76±0.06	9.69±0.16	13.51±0.4	7.12±0.11	8.08±1.44	1.85±0.04
Offline-LD (mQRDQN)	41.47±6.36	15.55±0.46	59.54±3.52	27.8±0.81	35.95±2.51	13.18±0.42	32.75±4.96	10.9±0.3	23.3±4.49	3.93±0.2
Offline-LD (d-mSAC)	20.78±5.21	6.75±1.63	28.37±0.97	14.84±0.77	29.33±1.33	12.93±0.52	21.76±2.85	7.76±0.64	14.73±2.45	4.35±0.52
IQL	21.12±4.65	7.59±0.57	28.71±2.85	16.24±0.43	26.89±2.31	11.63±0.56	22.18±2.55	7.64±0.4	13.96±1.01	3.14±0.82
<b>CDQAC</b>	<b>5.22±0.63</b>	<b>2.19±0.62</b>	<b>16.76±2.09</b>	<b>9.3±0.36</b>	<b>22.62±6.05</b>	<b>11.05±3.2</b>	<b>13.48±0.97</b>	<b>5.92±0.37</b>	<b>4.91±1.07</b>	<b>0.97±0.2</b>
PDR-GA										
BC	26.02±2.15	8.21±0.17	53.55±6.26	28.02±2.45	23.61±1.62	10.91±0.87	16.26±3.61	7.15±0.4	5.89±2.1	1.08±0.09
Offline-LD (mQRDQN)	27.62±9.83	15.0±0.21	29.47±8.62	25.59±0.29	30.82±5.5	12.62±0.29	24.68±4.86	10.51±0.12	17.36±5.16	3.6±0.14
Offline-LD (d-mSAC)	43.5±3.7	21.72±5.92	55.46±4.38	24.48±1.64	38.71±1.75	19.46±1.27	32.65±3.36	13.12±1.35	22.98±2.97	8.78±2.03
IQL	11.42±2.36	6.89±0.25	33.97±4.27	19.32±1.4	24.65±2.62	10.79±0.56	16.03±1.22	6.91±0.25	6.44±1.63	1.19±0.17
<b>CDQAC</b>	<b>5.01±0.28</b>	<b>2.31±0.36</b>	<b>15.34±1.11</b>	<b>8.9±0.59</b>	<b>17.79±5.04</b>	<b>9.17±1.49</b>	<b>12.3±1.61</b>	<b>5.57±0.43</b>	<b>4.07±0.91</b>	<b>0.83±0.24</b>
Random										
BC	19.65±1.43	15.33±0.08	35.81±9.72	27.02±0.37	23.84±2.46	12.29±0.1	20.23±0.37	10.47±0.1	11.38±0.95	3.51±0.12
Offline-LD (mQRDQN)	21.73±9.18	14.9±0.19	40.78±4.11	26.36±0.46	33.87±1.93	12.81±0.25	24.68±1.19	10.52±0.1	15.62±3.59	3.59±0.08
Offline-LD (d-mSAC)	11.59±3.69	4.79±1.46	22.09±3.25	11.7±1.28	28.51±2.45	13.37±1.85	21.7±3.27	9.18±1.85	13.07±3.76	3.7±2.14
IQL	14.0±4.05	10.08±0.93	32.8±4.58	20.08±0.72	31.66±2.55	13.17±0.61	24.87±3.32	9.63±0.39	13.81±3.39	3.53±0.5
<b>CDQAC</b>	<b>5.2±0.66</b>	<b>2.87±0.73</b>	<b>16.52±0.3</b>	<b>9.73±0.43</b>	<b>16.53±1.59</b>	<b>9.02±0.28</b>	<b>11.63±0.52</b>	<b>5.66±0.17</b>	<b>3.25±0.2</b>	<b>0.76±0.05</b>

that training on a diverse set of examples is critical for strong generalization performance in offline RL for FJSP.

### I.3 Additional Results JSP

Table 15: Results on JSP benchmarks for CDQAC  $10 \times 5$ , for all training datasets (PDR, GA, PDR-GA and Random). The mean and standard deviation of the gap (%) are reported from four different seeds. **Bold** indicates best result (lowest gap) for either the Greedy and Sampling (100 solutions) evaluation.

Instance Size	Greedy				Sampling				
	PDR	GA	PDR-GA	Random	PDR	GA	PDR-GA	Random	
Taillard	$15 \times 15$	$16.26 \pm 0.67$	$16.12 \pm 0.69$	$16.33 \pm 0.95$	<b><math>15.9 \pm 0.7</math></b>	$11.5 \pm 0.51$	$11.27 \pm 0.86$	$11.23 \pm 0.48$	<b><math>10.8 \pm 0.55</math></b>
	$20 \times 15$	$20.55 \pm 0.95$	$19.7 \pm 1.05$	<b><math>19.6 \pm 1.91</math></b>	$19.98 \pm 1.91$	$14.8 \pm 0.37$	$14.23 \pm 0.75$	$14.64 \pm 0.58$	<b><math>14.12 \pm 0.77</math></b>
	$20 \times 20$	$18.65 \pm 0.73$	$18.89 \pm 1.27$	$17.45 \pm 0.7$	<b><math>17.19 \pm 1.38</math></b>	<b><math>13.29 \pm 0.68</math></b>	$14.1 \pm 0.72$	$13.88 \pm 0.35$	$13.39 \pm 0.84$
	$30 \times 15$	$20.4 \pm 0.65$	$21.32 \pm 2.78$	$20.44 \pm 1.13$	<b><math>19.56 \pm 0.49</math></b>	$15.83 \pm 0.34$	$16.04 \pm 0.91$	$16.0 \pm 0.31$	<b><math>15.3 \pm 1.13</math></b>
	$30 \times 20$	$22.05 \pm 1.64$	$22.58 \pm 2.72$	<b><math>21.6 \pm 2.04</math></b>	$22.28 \pm 1.01$	<b><math>17.89 \pm 0.92</math></b>	$18.6 \pm 1.3$	$18.6 \pm 0.4$	$18.27 \pm 0.82$
	$50 \times 15$	$14.26 \pm 1.1$	$14.48 \pm 1.63$	$13.53 \pm 1.41$	<b><math>13.06 \pm 1.47</math></b>	$10.86 \pm 0.66$	<b><math>10.21 \pm 0.75</math></b>	$10.47 \pm 1.18$	$10.46 \pm 1.22$
	$50 \times 20$	$14.46 \pm 0.95$	$15.21 \pm 3.36$	<b><math>13.83 \pm 1.18</math></b>	$13.9 \pm 1.3$	$11.6 \pm 0.35$	$12.07 \pm 1.21$	$11.62 \pm 0.47$	<b><math>11.37 \pm 0.52</math></b>
	$100 \times 20$	$6.43 \pm 0.12$	$8.1 \pm 4.73$	$6.18 \pm 0.82$	<b><math>5.53 \pm 1.12</math></b>	$4.66 \pm 0.14$	$4.46 \pm 1.33$	$4.56 \pm 0.49$	<b><math>4.25 \pm 0.59</math></b>
Mean	$16.63 \pm 0.85$	$17.05 \pm 2.28$	$16.12 \pm 1.27$	<b><math>15.93 \pm 1.17</math></b>	$12.55 \pm 0.5$	$12.62 \pm 0.98$	$12.62 \pm 0.53$	<b><math>12.24 \pm 0.8</math></b>	
Demirkol	$20 \times 15$	$24.87 \pm 1.51$	<b><math>24.03 \pm 0.94</math></b>	$24.47 \pm 2.11$	$24.49 \pm 1.83$	$19.4 \pm 0.63$	$19.29 \pm 0.94$	$19.63 \pm 0.81$	<b><math>18.82 \pm 0.86</math></b>
	$20 \times 20$	$23.3 \pm 0.36$	<b><math>21.29 \pm 1.19</math></b>	$22.01 \pm 1.12$	$21.71 \pm 1.47$	$17.66 \pm 0.45$	$17.62 \pm 1.15$	$18.03 \pm 0.54$	<b><math>17.13 \pm 0.71</math></b>
	$30 \times 15$	$29.63 \pm 0.69$	<b><math>28.22 \pm 1.8</math></b>	$28.71 \pm 2.63$	$28.76 \pm 1.72$	$24.21 \pm 0.61$	<b><math>23.22 \pm 1.1</math></b>	$24.2 \pm 1.21$	$23.67 \pm 1.7$
	$30 \times 20$	$28.72 \pm 1.13$	<b><math>28.33 \pm 1.0</math></b>	$28.53 \pm 2.57$	$28.6 \pm 2.39$	$23.72 \pm 0.61$	$23.71 \pm 0.5$	$24.15 \pm 1.55$	<b><math>23.56 \pm 1.29</math></b>
	$40 \times 15$	$26.98 \pm 1.0$	<b><math>25.1 \pm 1.35</math></b>	$25.76 \pm 2.78$	$25.51 \pm 2.85$	$22.62 \pm 0.98$	<b><math>20.31 \pm 0.84</math></b>	$21.73 \pm 1.63$	$21.15 \pm 1.66$
	$40 \times 20$	$29.42 \pm 1.18$	<b><math>27.49 \pm 1.45</math></b>	$28.5 \pm 2.67$	$28.77 \pm 1.74$	$24.88 \pm 0.18$	<b><math>24.06 \pm 1.03</math></b>	$25.1 \pm 1.7$	$24.58 \pm 1.49$
	$50 \times 15$	$27.82 \pm 0.94$	<b><math>25.03 \pm 2.61</math></b>	$26.49 \pm 3.84$	$25.06 \pm 5.42$	$23.8 \pm 0.85$	<b><math>20.83 \pm 0.97</math></b>	$22.53 \pm 2.5$	$22.5 \pm 2.74$
	$50 \times 20$	$30.43 \pm 0.96$	<b><math>27.5 \pm 1.63</math></b>	$28.71 \pm 2.98$	$28.65 \pm 2.58$	$26.35 \pm 0.69$	<b><math>24.65 \pm 1.28</math></b>	$26.1 \pm 1.52$	$25.67 \pm 1.06$
Mean	$27.65 \pm 0.97$	<b><math>25.87 \pm 1.5</math></b>	$26.65 \pm 2.59$	$26.44 \pm 2.5$	$22.83 \pm 0.62$	<b><math>21.71 \pm 0.98</math></b>	$22.68 \pm 1.43$	$22.13 \pm 1.44$	

Table 16: Results on JSP benchmarks for CDQAC  $15 \times 10$ , for all training datasets (PDR, GA, PDR-GA and Random). The mean and standard deviation of the gap (%) are reported from four different seeds. **Bold** indicates best result (lowest gap) for either the Greedy and Sampling (100 solutions) evaluation.

Instance Size	Greedy				Sampling				
	PDR	GA	PDR-GA	Random	PDR	GA	PDR-GA	Random	
Taillard	$15 \times 15$	$16.73 \pm 0.6$	$17.23 \pm 1.28$	$17.35 \pm 1.89$	<b><math>16.7 \pm 0.97</math></b>	$11.6 \pm 0.48$	$11.26 \pm 0.84$	$11.39 \pm 0.86$	<b><math>11.24 \pm 0.83</math></b>
	$20 \times 15$	$21.63 \pm 0.33$	$21.4 \pm 1.92$	$21.57 \pm 2.04$	<b><math>20.69 \pm 1.09</math></b>	$15.22 \pm 0.23$	$14.77 \pm 1.13$	$14.87 \pm 0.92$	<b><math>14.71 \pm 0.28</math></b>
	$20 \times 20$	$18.73 \pm 0.71$	$18.51 \pm 1.41$	$19.0 \pm 1.11$	<b><math>18.14 \pm 0.81</math></b>	$13.61 \pm 0.59$	<b><math>13.49 \pm 0.57</math></b>	$13.76 \pm 0.52$	$13.53 \pm 0.5$
	$30 \times 15$	<b><math>20.6 \pm 0.65</math></b>	$21.27 \pm 1.27$	$21.33 \pm 1.4$	$20.86 \pm 0.88$	$16.01 \pm 0.14$	$16.21 \pm 0.88$	$16.07 \pm 0.51$	<b><math>15.92 \pm 0.3</math></b>
	$30 \times 20$	$23.52 \pm 1.14$	<b><math>23.17 \pm 0.34</math></b>	$23.94 \pm 0.69$	$23.55 \pm 1.14$	$18.43 \pm 0.6$	<b><math>18.15 \pm 0.47</math></b>	$18.43 \pm 0.62$	$18.29 \pm 0.5$
	$50 \times 15$	$14.9 \pm 0.28$	$14.6 \pm 1.09$	<b><math>14.05 \pm 1.31</math></b>	$15.47 \pm 2.47$	$11.45 \pm 0.54$	$10.71 \pm 1.06$	$10.8 \pm 1.4$	<b><math>10.48 \pm 0.43</math></b>
	$50 \times 20$	<b><math>14.82 \pm 0.77</math></b>	$16.46 \pm 0.99$	$15.41 \pm 1.03$	$16.47 \pm 4.19$	$12.05 \pm 0.54$	$11.93 \pm 0.82$	$12.17 \pm 1.13$	<b><math>11.57 \pm 0.24</math></b>
	$100 \times 20$	$6.44 \pm 0.34$	$8.24 \pm 2.43$	<b><math>6.01 \pm 0.97</math></b>	$8.0 \pm 3.19$	$4.88 \pm 0.25$	$4.73 \pm 0.34$	<b><math>4.52 \pm 0.49</math></b>	$4.96 \pm 0.75$
Mean	<b><math>17.17 \pm 0.6</math></b>	$17.61 \pm 1.34$	$17.33 \pm 1.31$	$17.48 \pm 1.84$	$12.91 \pm 0.42$	$12.66 \pm 0.77$	$12.75 \pm 0.81$	<b><math>12.59 \pm 0.48</math></b>	
Demirkol	$20 \times 15$	$27.13 \pm 0.74$	$26.03 \pm 1.0$	<b><math>24.94 \pm 1.91</math></b>	$26.05 \pm 1.37$	$20.23 \pm 0.8$	<b><math>19.4 \pm 1.05</math></b>	$19.5 \pm 1.3$	$19.59 \pm 0.72$
	$20 \times 20$	$24.01 \pm 0.5$	$22.86 \pm 1.19$	$22.73 \pm 2.13$	<b><math>22.67 \pm 1.4</math></b>	$17.59 \pm 0.62$	$17.4 \pm 0.62$	$17.6 \pm 0.66$	<b><math>17.23 \pm 0.68</math></b>
	$30 \times 15$	$30.3 \pm 1.13$	$29.66 \pm 1.54$	$29.19 \pm 1.49$	<b><math>29.15 \pm 1.19</math></b>	$25.93 \pm 1.37$	<b><math>24.04 \pm 1.21</math></b>	$24.05 \pm 1.9$	$24.25 \pm 0.87$
	$30 \times 20$	$30.43 \pm 1.04$	$28.65 \pm 1.32$	$28.5 \pm 2.35$	<b><math>28.24 \pm 1.57</math></b>	$24.92 \pm 0.64$	<b><math>23.0 \pm 0.83</math></b>	$23.72 \pm 1.55$	$23.46 \pm 1.07$
	$40 \times 15$	$27.81 \pm 0.97$	$25.68 \pm 0.97$	<b><math>24.77 \pm 2.6</math></b>	$25.61 \pm 1.71$	$23.51 \pm 1.04$	<b><math>21.03 \pm 1.25</math></b>	$21.08 \pm 2.15$	$21.38 \pm 1.49$
	$40 \times 20$	$30.54 \pm 1.26$	<b><math>27.64 \pm 2.05</math></b>	$28.47 \pm 2.71$	$28.99 \pm 0.81$	$25.86 \pm 1.0$	<b><math>23.63 \pm 0.98</math></b>	$24.48 \pm 1.73$	$24.21 \pm 1.6$
	$50 \times 15$	$29.14 \pm 0.94$	<b><math>23.84 \pm 4.59</math></b>	$24.28 \pm 5.27$	$26.16 \pm 2.21$	$25.09 \pm 1.04$	$20.78 \pm 2.54$	$21.87 \pm 3.35$	<b><math>20.65 \pm 3.01</math></b>
	$50 \times 20$	$31.56 \pm 1.3$	$28.85 \pm 1.36$	<b><math>28.43 \pm 1.44</math></b>	$30.53 \pm 1.94$	$27.19 \pm 1.34$	<b><math>24.4 \pm 0.79</math></b>	$24.97 \pm 0.87$	$25.47 \pm 1.48$
Mean	$28.87 \pm 0.99$	$26.65 \pm 1.75$	<b><math>26.42 \pm 2.49</math></b>	$27.18 \pm 1.52$	$23.79 \pm 0.98$	<b><math>21.71 \pm 1.16</math></b>	$22.16 \pm 1.69$	$22.03 \pm 1.36$	

In Sect. 5.4, we compared CDQAC on the Taillard and Demirkol instances. The results in Table 7 included only CDQAC trained on the Random dataset for  $10 \times 5$  instances. In this section, we show the results for the other training sets for both  $10 \times 5$  (Table 15) and  $15 \times 10$  (Table 16) instances.

Tables 15 and 16 show only minor performance differences between the training datasets. Table 16 contains the largest difference between the mean Greedy results of Demirkol between PDR ( $28.87\% \pm 0.99\%$ ) and PDR-GA ( $26.42\% \pm 2.49\%$ ). We also notice that PDR and Random perform better with the Taillard instances compared to GA, but GA performs better on the Demirkol instances. We hypothesize that this difference comes from the differing distributions of processing times: Demirkol instances have processing times ranging from 1 to 200 and those of Taillard only from 1 to 100, whereby CDQAC was trained on instances similar to

Taillard instances. These results contrast with those of FJSP in App. I.2, where GA was unable to generalize well to benchmark instances that have a different distribution to the training instances. These results suggest that the choice of training data has a fundamentally different impact in JSP compared to FJSP.

#### I.4 Additional Results Dataset Size

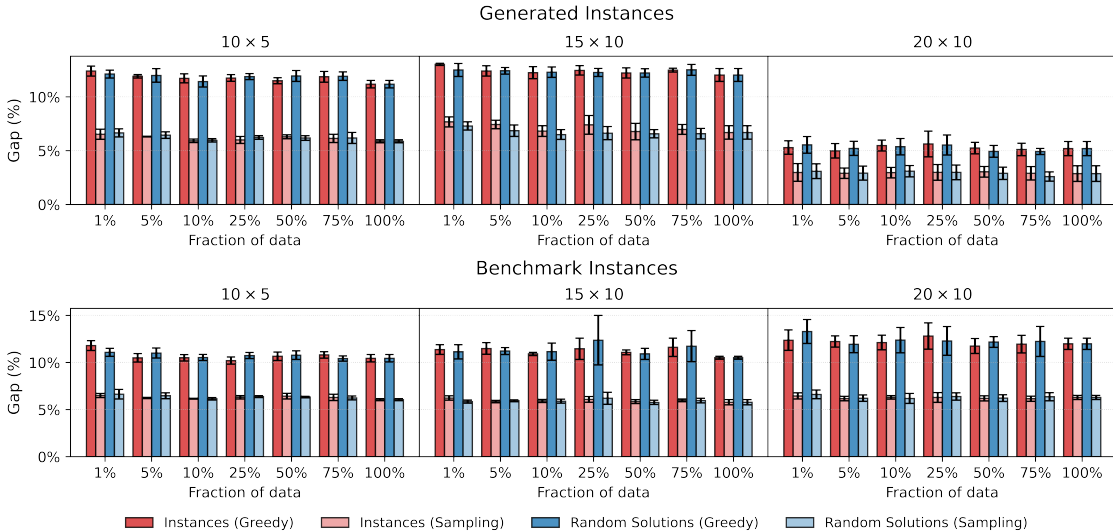


Figure 8: Effect of different dataset sizes. We evaluate the sample efficiency of CDQAC by reducing the Random training dataset in two ways. **Red**: the number of *instances* (1%: 5 instances, 5%: 25 instances, 10%: 50 instances, 25%: 125 instances, 50%: 250 instances, 75%: 375 instances, 100%: 500 instances, with each instance having 100 random solutions). **Blue**: the number of *random solutions* per instance (1%: 1 solution, 5%: 5 solutions, 10%: 10 solutions, 25%: 25 solutions, 50%: 50 solutions, 75%: 75 solutions, 100%: 100 solutions, for each instance, with 500 instances in total). Performance is reported as the mean gap across four seeds, with error bars indicating standard deviation.

In Sect. 5.5, we demonstrated that reducing the number of training in the Random training dataset had little impact on overall performance on the FJSP benchmark sets, Brandimarte and Hurink. In this section, we provide a more comprehensive analysis by including results on generated evaluation instances. Additionally, we introduce a second evaluation for the reduction of the dataset, in which we decrease the number of solutions generated per instance by the random policy. For both evaluations, we considered subsets containing 1%, 5%, 10%, 25%, 50%, 75%, and 100% of the original dataset size. Specifically, when reducing the number of instances, we used either 5, 25, 50, 125, 250, 375, or 500 instances, each with 100 random solutions. When reducing the number of random solutions per instance, we used 500 instances, each with either 1, 5, 10, 25, 50, 75, or 100 random solutions.

As shown in Fig. 8, decreasing the dataset, either by limiting the number of instances or by reducing the number of random solutions per instance, does not lead to a significant loss in performance. The results remain relatively stable, with the standard deviation mostly below 1.5%. The sole exception occurs for  $15 \times 10$  on the benchmark instances at 25%, when reducing the number of random solutions, where the greedy evaluation shows a standard deviation of 2.63%. Notably, this increased standard deviation is only observed for benchmark instances and not for generated instances at 25% random solutions, as evidenced in Fig. 8. This suggests that larger datasets may improve generalization to previously unseen instances. Another benefit is training stability, with larger dataset producing a smaller standard deviation. In general, these findings reinforce our conclusion from Sect. 5.5: CDQAC maintains competitive performance even when trained on substantially reduced datasets, underscoring its sample efficiency.

Table 17: Results JSP benchmarks. Average gap (%) is reported. In this additional comparison, we compare CDQAC to constructive learning-based approaches that only function for JSP and to approaches that function for both JSP and FJSP. The approaches that only function for JSP are: L2D (Zhang et al., 2020), CL (Iklassov et al., 2023), Sched (Park et al., 2021), SL (Corsini et al., 2024), GD (Pirnay & Grimm, 2024), and OD (Remmerden et al., 2025). Approaches that can do both JSP and FJSP are: DAN (Wang et al., 2023), Res (Ho et al., 2024), and CDQAC (ours). We additionally include the JSP-only improvement heuristics L2S (Zhang et al., 2024a) and TBGAT (Zhang et al., 2024b) evaluated for 500 and 1000 improvement steps. We note the best performing overall approach with \*, and the best approach that can handle both JSP and FJSP in **bold**.

		Greedy								Sampling						Improvement				
		L2D	CL	Sched	SL	GD	OD	DAN	Res	CDQAC	CL <sup>a</sup>	SL <sup>a</sup>	GD <sup>b</sup>	DAN <sup>c</sup>	Res <sup>c</sup>	CDQAC <sup>c</sup>	L2S		TBGAT	
Instance Size																	500	1000	500	1000
Taillard	15 × 15	28.1	14.3	15.3	13.8	9.6	25.8	19.0	17.6	<b>15.0</b>	9.0	7.2*	10.1	13.2	13.3	<b>10.4</b>	9.3	8.6	8.0	6.1
	20 × 15	32.7	16.5	19.4	15.0	9.9*	30.2	22.1	21.2	<b>17.7</b>	10.6	9.3*	9.8	17.4	16.1	<b>13.2</b>	11.6	10.4	9.9	8.7
	20 × 20	31.8	17.3	17.2	15.2	11.1*	28.9	18.0	18.0	<b>17.6</b>	10.9	10.0*	10.4	13.3	15.8	<b>12.9</b>	12.4	11.4	10.0	9.0
	30 × 15	30.2	18.5	18.0	17.1	9.5*	29.2	21.7	20.1	<b>19.1</b>	14.0	11.0	8.5*	17.2	18.0	<b>14.9</b>	14.7	12.9	13.3	10.9
	30 × 20	35.2	21.5	18.7	18.5	13.8*	33.1	23.2	22.3	<b>21.2</b>	16.1	13.4	12.3*	19.0	19.7	<b>17.9</b>	17.5	15.7	16.4	14.0
	50 × 15	21.0	12.2	13.8	10.1	2.7*	20.6	14.8	15.6	<b>13.0</b>	9.3	5.5	2.6*	12.7	13.2	<b>9.9</b>	11.0	9.0	9.6	7.5
	50 × 20	26.1	13.2	13.5	11.6	6.7*	24.3	16.0	14.4	<b>12.8</b>	9.9	8.4	7.7*	13.1	14.1	<b>11.0</b>	13.0	11.4	11.9	9.4
100 × 20	13.3	5.9	6.6	5.8	1.7*	12.7	7.3	6.5	<b>5.3</b>	4.0	2.3	1.3*	5.9	6.5	<b>3.6</b>	7.9	6.6	6.4	4.9	
Mean		27.3	14.9	15.4	13.4	8.1*	25.6	18.2	17.0	<b>15.2</b>	10.5	8.4	7.8*	14.4	14.6	<b>11.7</b>	12.2	10.8	10.7	8.8
Demirkol	20 × 15	36.3	-	-	18.0*	-	35.8	-	26.1	<b>22.9</b>	-	12.0*	-	-	22.6	<b>18.4</b>	-	-	-	-
	20 × 20	34.4	-	-	19.4*	-	32.8	-	21.5	<b>20.3</b>	-	13.5*	-	-	18.9	<b>16.5</b>	-	-	-	-
	30 × 15	37.8	-	-	21.8*	-	38.8	-	27.6	<b>27.1</b>	-	14.4*	-	-	29.4	<b>23.1</b>	-	-	-	-
	30 × 20	38.0	-	-	25.7*	-	36.0	-	29.9	<b>27.9</b>	-	17.1*	-	-	28.3	<b>23.4</b>	-	-	-	-
	40 × 15	34.6	-	-	17.5*	-	35.5	-	26.2	<b>25.5</b>	-	11.7*	-	-	28.4	<b>20.2</b>	-	-	-	-
	40 × 20	39.2	-	-	22.2*	-	38.5	-	27.7	<b>27.9</b>	-	16.0*	-	-	30.9	<b>24.1</b>	-	-	-	-
	50 × 15	33.2	-	-	15.7*	-	34.1	-	27.4	<b>25.0</b>	-	11.2*	-	-	29.5	<b>21.7</b>	-	-	-	-
50 × 20	37.7	-	-	22.4*	-	38.9	-	30.0	<b>28.6</b>	-	15.8*	-	-	32.8	<b>25.1</b>	-	-	-	-	
Mean		36.4	-	-	20.3*	-	36.3	-	27.0	<b>25.7</b>	-	14.0*	-	-	27.6	<b>21.6</b>	-	-	-	-

<sup>a</sup> Used 128 samples for each instance during the sampling evaluation.

<sup>b</sup> Used beam search with a width of 16.

<sup>c</sup> Used 100 samples for each instance during the sampling evaluation.

## 1.5 Additional JSP baselines

Table 17 provides an extended JSP comparison that includes specialized JSP-only baselines excluded from Table 7, where the requirement was that methods must operate on both JSP and FJSP. Among constructive RL approaches, to which CDQAC also belongs, CDQAC is competitive with CL (Iklassov et al., 2023) and Sched (Park et al., 2021), with CL slightly ahead overall. The self-labeling methods SL (Corsini et al., 2024) and GD (Pirnay & Grimm, 2024) achieve lower gaps, and the improvement-based methods L2S (Zhang et al., 2024a) and TBGAT (Zhang et al., 2024b) outperform CDQAC on smaller Taillard instances. CDQAC scales more favorably, however: on the largest Taillard instances (100 × 20), its sampling evaluation (3.6%) outperforms both L2S-1000 (6.6%) and TBGAT-1000 (4.9%).

These comparisons should be read in context. All additional baselines are JSP-specific and require a training environment, so none extend to FJSP, where CDQAC remains state-of-the-art. The lower gaps reported by SL and GD also come at substantial training cost: both methods require up to seven days of GPU training, while improvement methods incur additional inference cost proportional to the number of improvement steps. CDQAC, by contrast, trains in one to two hours from a static random dataset, without simulator access or expert demonstrations. We therefore position CDQAC not as a replacement for specialized JSP methods but as a complementary offline approach with broader applicability and substantially lower training cost.

Table 18: Multi-seed comparison on FJSP benchmark sets (Brandimarte and Hurink). Both methods trained on the Random dataset at the indicated training size. Single-seed DANIEL values are taken from Table 4 of the main paper for reference. Mean and standard deviation of the gap (%) reported over four seeds. Bold indicates the lower mean gap among the multi-seed entries within each (training size, evaluation mode) row group.

	Train size	Method	Brandimarte (mk)	Hurink edata	Hurink rdata	Hurink vdata
Greedy	$10 \times 5$	DANIEL (single seed) <sup>†</sup>	13.58	16.33	11.42	3.28
		DANIEL (multi-seed)	$14.04 \pm 1.44$	$15.72 \pm 0.47$	$11.24 \pm 0.60$	<b><math>2.94 \pm 0.12</math></b>
		CDQAC (Ours)	<b><math>13.78 \pm 0.78</math></b>	<b><math>14.53 \pm 0.41</math></b>	<b><math>10.40 \pm 0.36</math></b>	$3.10 \pm 0.22$
	$15 \times 10$	DANIEL (single seed) <sup>†</sup>	12.97	14.41	12.07	3.75
		DANIEL (multi-seed)	<b><math>13.58 \pm 0.24</math></b>	$14.97 \pm 0.14$	$11.15 \pm 0.37$	<b><math>3.06 \pm 0.07</math></b>
		CDQAC (Ours)	<b><math>13.58 \pm 0.66</math></b>	<b><math>14.56 \pm 0.55</math></b>	<b><math>10.77 \pm 0.36</math></b>	$3.16 \pm 0.10$
Sampling	$10 \times 5$	DANIEL (single seed) <sup>†</sup>	9.53	9.08	4.95	0.69
		DANIEL (multi-seed)	$9.66 \pm 0.63$	<b><math>8.90 \pm 0.55</math></b>	$5.34 \pm 0.13$	<b><math>0.66 \pm 0.06</math></b>
		CDQAC (Ours)	<b><math>8.67 \pm 0.21</math></b>	$9.54 \pm 0.39$	<b><math>5.30 \pm 0.22</math></b>	$0.68 \pm 0.03$
	$15 \times 10$	DANIEL (single seed) <sup>†</sup>	8.95	8.72	5.49	0.72
		DANIEL (multi-seed)	$8.89 \pm 0.46$	$9.47 \pm 0.47$	$5.44 \pm 0.24$	<b><math>0.64 \pm 0.05</math></b>
		CDQAC (Ours)	<b><math>8.73 \pm 0.73</math></b>	<b><math>8.51 \pm 0.52</math></b>	<b><math>5.22 \pm 0.12</math></b>	$0.67 \pm 0.02$

<sup>†</sup> As reported in Table 4 of the main paper.

## J Significance Test

Our comparison for FJSP (Sect. 5.3) and JSP (Sect. 5.4) showed that CDQAC outperformed DANIEL (Wang et al., 2023) in most evaluations. To assess whether these results are significant, we conducted a one-sided Wilcoxon signed-rank test for both JSP and FJSP.

### J.1 FJSP.

Although CDQAC consistently outperformed DANIEL in most FJSP evaluations, the margins were smaller than in other results. To this end, we paired all results from Tables 4, 5, and 6, in both greedy and sampling evaluations. Furthermore, we paired the results of both  $10 \times 5$  and  $15 \times 10$  in Table 4, resulting in a sample size of 26 pairs. The statistical test yielded a  $p \approx 0.018$  rejecting the null hypothesis of  $p > 0.05$ , indicating that CDQAC, trained solely on random data, significantly outperforms the online RL baseline DANIEL (Wang et al., 2023) in our FJSP evaluation.

**Multi-seed Comparison with DANIEL** The Wilcoxon test above relies on the single-seed DANIEL results in Tables 4, 5, and 6, following the convention of prior online RL work for FJSP, which typically reports single-seed results (see Sect. 5.3). To verify that the observed significance is not an artifact of an unfavorable seed choice for DANIEL, we additionally retrain DANIEL (Wang et al., 2023) with the same four random seeds as CDQAC (1, 2, 3, 4) Tables 18, 19, and 20 report the multi-seed results, per benchmark set, for CDQAC and DANIEL Wang et al. (2023) and the single-seed results of DANIEL, as reference, originally given in Tables 4, 5, and 6, respectively.

The single-seed DANIEL values fall within approximately one to two standard deviations of the multi-seed results, indicating that the originally reported values are representative of DANIEL performance. Across the 16 benchmark results in Table 18, CDQAC achieves the lower mean gap on 10 results, ties on 1, and trails on 5, while on the four large-instance generalization sets in Table 20 CDQAC outperforms DANIEL on all four by 1.3–2.8 percentage points; DANIEL retains its edge on the largest in-distribution training size ( $20 \times 10$  in Table 19), consistent with the observation in Sect. 5.3 that online RL is most competitive within its training distribution. Repeating the one-sided Wilcoxon signed-rank test on the same 26 pairs using the multi-seed DANIEL means yields  $p \approx 0.03$ , which still rejects the null hypothesis at the 0.05 level. The conclusion that CDQAC significantly outperforms DANIEL on FJSP therefore holds under multi-seed evaluation.

Table 19: Multi-seed comparison on generated FJSP evaluation instances. Both methods are trained and evaluated at the same instance size, on the Random dataset. Single-seed DANIEL values are taken from Table 5 of the main paper for reference. Mean and standard deviation of the gap (%) reported over four seeds. Bold indicates the lower mean gap among the multi-seed entries.

Method		$10 \times 5$	$15 \times 10$	$20 \times 10$
Greedy	DANIEL (single seed) <sup>†</sup>	10.87	12.42	1.31
	DANIEL (multi-seed)	$12.14 \pm 0.61$	$12.22 \pm 0.48$	<b><math>1.98 \pm 0.62</math></b>
	CDQAC (Ours)	<b><math>11.19 \pm 0.35</math></b>	<b><math>12.04 \pm 0.59</math></b>	$5.20 \pm 0.66$
Sampling	DANIEL (single seed) <sup>†</sup>	5.57	6.79	-1.03
	DANIEL (multi-seed)	$6.18 \pm 0.36$	<b><math>6.46 \pm 0.55</math></b>	<b><math>-0.18 \pm 0.53</math></b>
	CDQAC (Ours)	<b><math>5.87 \pm 0.14</math></b>	$6.70 \pm 0.62$	$2.87 \pm 0.73$

<sup>†</sup> As reported in Table 5 of the main paper.

Table 20: Multi-seed comparison on large generalization FJSP instances. Both methods trained on  $10 \times 5$  instances with the Random dataset, evaluated on larger unseen sizes. Single-seed DANIEL values are taken from Table 6 of the main paper for reference. Mean and standard deviation of the gap (%) reported over four seeds. Bold indicates the lower mean gap among the multi-seed entries.

Method		$30 \times 10$	$40 \times 10$
Greedy	DANIEL (single seed) <sup>†</sup>	5.10	3.65
	DANIEL (multi-seed)	$6.63 \pm 0.88$	$5.19 \pm 0.91$
	CDQAC (Ours)	<b><math>5.37 \pm 0.68</math></b>	<b><math>3.89 \pm 0.68</math></b>
Sampling	DANIEL (single seed) <sup>†</sup>	4.43	3.77
	DANIEL (multi-seed)	$6.30 \pm 0.90$	$5.70 \pm 0.92$
	CDQAC (Ours)	<b><math>3.79 \pm 0.61</math></b>	<b><math>2.86 \pm 0.56</math></b>

<sup>†</sup> As reported in Table 6 of the main paper.

## J.2 JSP.

To evaluate the significance of the JSP results, we again paired the results of CDQAC and DANIEL in Table 7, whereby we paired each Taillard result, both for greedy and sampling. This results in a sample size of 16 pairs. The Wilcoxon test resulted in  $p \approx 0.00022$ , indicating that CDQAC also significantly outperforms DANIEL on JSP.