Fair Resource Allocation in Weakly Coupled Markov Decision Processes

Xiaohui Tu *[1,3], Yossiri Adulyasak[2], Nima Akbarzadeh[1,3], and Erick Delage[1,3]

[1]*GERAD & Department of Decision Sciences, HEC Montréal, Montréal, H3T2A7, Canada*
[2]*GERAD & Department of Logistics and Operations Management, HEC Montréal, Montréal, H3T2A7, Canada*
[3]*MILA - Quebec AI Institute, Montréal, H2S3H1, Canada*

November 1, 2024

Abstract

We consider fair resource allocation in sequential decision-making environments modeled as weakly coupled Markov decision processes, where resource constraints couple the action spaces of N sub-Markov decision processes (sub-MDPs) that would otherwise operate independently. We adopt a fairness definition using the generalized Gini function instead of the traditional utilitarian (total-sum) objective. After introducing a general but computationally prohibitive solution scheme based on linear programming, we focus on the homogeneous case where all sub-MDPs are identical. For this case, we show for the first time that the problem reduces to optimizing the utilitarian objective over the class of "permutation invariant" policies. This result is particularly useful as we can exploit Whittle index policies in the restless bandits setting while, for the more general setting, we introduce a count-proportion-based deep reinforcement learning approach. Finally, we validate our theoretical findings with comprehensive experiments, confirming the effectiveness of our proposed method in achieving fairness.

1 Introduction

Machine learning (ML) algorithms play a significant role in automated decision-making processes, influencing our daily lives. Mitigating biases within the ML pipeline is crucial to ensure fairness and generate reliable outcomes (Caton and Haas, 2024). Extensive research has been conducted to enhance fairness across various applications, such as providing job matching services (van den Broek et al., 2020; Raghavan et al., 2020), assigning credit scores and loans (Kozodoi et al., 2022), and delivering healthcare services (Farnadi et al., 2021; Chen et al., 2023).

However, most real-world decision processes are sequential in nature and past decisions may have an impact on equity. For example, if people are unfairly denied credit or job opportunities early in their careers, it can have long-term consequences for their financial well-being and opportunities for advancement (Liu et al., 2018). In addition, the impact of decisions can accumulate, which may lead to large disparities over time and can actually exacerbate unfairness (Creager et al., 2020; D'Amour et al., 2020).

Another motivating example for this work is taxi dispatching: when certain areas are consistently prioritized over others, it can lead to long-term disparities in service accessibility. This may lead to long waiting times for passengers in certain neighborhoods, while taxis run empty and seek passengers in other areas (Liu et al., 2021). Thus, to ensure that individuals are not treated unfairly based on past decisions or changing circumstances, we should extend the current definitions of fairness to consider the impact of feedback effects (Ghalme et al., 2022a).

Fairness is a complex and multi-faceted concept, and there are many different ways in which it can be operationalized and measured. We resort to the generalized Gini social welfare function (GGF) (Weymark, 1981), which

*Corresponding author. Email addresses: *xiaohui.tu@hec.ca*, *yossiri.adulyasak@hec.ca*, *nima.akbarzadeh@hec.ca*, and *erick.delage@hec.ca*

covers various fairness measures as special cases. The long-term impacts of fair decision dynamics have recently been approached using Markov decision processes (MDPs) (Wen et al., 2021; Puranik et al., 2022; Ghalme et al., 2022b). By studying fairness in MDPs, we can better understand how such considerations can be incorporated into decision-making processes, how they may affect the outcomes of those processes, and how to develop approaches to promote fairness in a wider range of settings.

To the best of our knowledge, we are the first to incorporate fairness considerations in the form of the GGF objective within weakly coupled Markov decision processes (WCMDPs) (Hawkins, 2003; Adelman and Mersereau, 2008), which can also be considered as multi-action multi-resource restless multi-arm bandits. This model captures the complex interactions among coupled MDPs over time and allows the applicability of our work to various applications in scheduling (Saure et al., 2012; El Shar and Jiang, 2024), application screening (Gast et al., 2024), budget allocation (Boutilier and Lu, 2016), inventory (El Shar and Jiang, 2024), and restless multi-arm bandits (RMABs) (Hawkins, 2003; Zhang, 2022).

Contributions Our contributions are as follows. *Theoretically*, we reformulate the WCMDP problem with the GGF objective as a linear programming (LP) problem, and show that, under symmetry, it reduces to maximizing the average expected total discounted rewards, called the utilitarian approach. *Methodologically*, we propose a state count approach to further simplify the problem, and introduce a count proportion-based deep reinforcement learning (RL) method that can solve the reduced problem efficiently and can scale to larger cases by assigning resources proportionally to the number of stakeholders. *Experimentally*, we design various experiments to show the GGF-optimality, flexibility, scalability and efficiency of the proposed deep RL approach. We benchmark our approach against the Whittle index policy on machine replacement applications modeled as RMABs (Akbarzadeh and Mahajan, 2019), showing the effectiveness of our method in achieving fair outcomes under different settings.

There are two studies closely related to our work. The first work by Gast et al. (2024) considers symmetry simplification and count aggregation MDPs. They focus on solving a LP model repeatedly with a total-sum objective to obtain asymptotic optimal solutions when the number of coupled MDPs is very large, whereas we explicitly address the fairness aspect and exploit a state count representation to design scalable deep RL approaches. The second related work by Siddique et al. (2020) integrates the fair Gini multi-objective RL to treat every user equitably. This fair optimization problem is later extended to the decentralized cooperative multi-agent RL by Zimmer et al. (2021), and further refined to incorporate preferential treatment with human feedback by Siddique et al. (2023) and Yu et al. (2023). In contrast, our work demonstrates that the WCMDP with the GGF objective and identical coupled MDPs reduces to a much simpler utilitarian problem, which allows us to exploit its structure to develop efficient and scalable algorithms. A more comprehensive literature review on fairness in MDPs, RL, and RMABs, is provided in Appendix A to clearly position our work.

2 Background

We review infinite-horizon WCMDPs and introduce the GGF for encoding fairness. We then define the fair optimization problem and provide an exact solution scheme based on linear programming.

Notation Let $[N] := \{1, \ldots, N\}$ for any integer N. For any vector $\boldsymbol{v} \in \mathbb{R}^N$, the n-th element is denoted as v_n and the average value as $\bar{v} = \frac{1}{N}\sum_{n=1}^{N} v_n$. An indicator function $\mathbb{I}\{x \in A\}$ equals 1 if $x \in A$ and 0 otherwise. For any set X, $\Delta(X)$ represents the set of all probability distributions over X. We let $\mathbb{S}^N$ be the set of all $N!$ permutations of the indices in $[N]$ and $\mathcal{G}^N$ be the set of all permutation operators so that $Q \in \mathcal{G}^N$ if and only if there exists a $\sigma \in \mathbb{S}^N$ such that $Q\boldsymbol{v}(n) = \boldsymbol{v}_{\sigma(n)}$ for all $n \in [N]$ when $\boldsymbol{v} \in \mathbb{R}^N$.

2.1 The Weakly Coupled MDP

We consider N MDPs indexed by $n \in \mathcal{N} := [N]$ interacting in discrete-time over an infinite horizon $t \in \mathcal{T} := \{0, 1, \ldots, \infty\}$. The n-th MDP $\mathcal{M}_n$, also referred as sub-MDP, is defined by a tuple $(\mathcal{S}_n, \mathcal{A}_n, p_n, r_n, \mu_n, \gamma)$, where $\mathcal{S}_n$ is a finite set of states with cardinality S, and $\mathcal{A}_n$ is a finite set of actions with cardinality A. The transition probability function is defined as $p_n(s'_n|s_n, a_n) = \mathbb{P}(s_{t+1,n} = s'_n|s_{t,n} = s_n, a_{t,n} = a_n)$, which represents the probability of reaching state $s'_n \in \mathcal{S}_n$ after performing action $a_n \in \mathcal{A}_n$ in state $s_n \in \mathcal{S}_n$ at time t. The reward function $r_n(s_n, a_n)$ denotes the immediate real-valued reward obtained by executing action a_n in state s_n. Although the transition probabilities and the reward function may vary with the sub-MDP n, we assume that they are stationary

across all time steps for simplicity. The initial state distribution is represented by $\mu_n \in \Delta(\mathcal{S}_n)$, and the discount factor, common to all sub-MDPs, is denoted by $\gamma \in [0, 1)$.

An infinite-horizon WCMDP $\mathcal{M}^{(N)}$ consists of N sub-MDPs, where each sub-MDP is independent of the others in terms of state transitions and rewards. They are linked to each other solely through a set of K constraints on their actions at each time step. Formally, the WCMDP is defined by a tuple $(\mathcal{S}^{(N)}, \mathcal{A}^{(N)}, p^{(N)}, \boldsymbol{r}, \mu^{(N)}, \gamma)$, where the state space $\mathcal{S}^{(N)}$ is the Cartesian product of individual state spaces, and the action space $\mathcal{A}^{(N)}$ is a subset of the Cartesian product of action spaces, defined as $\mathcal{A}^{(N)} := \{(a_1, \ldots, a_N) \mid \sum_{n=1}^{N} d_{k,n}(a_n) \leq b_k, \forall k \in \mathcal{K},\ a_n \in \mathcal{A}_n, \forall n \in \mathcal{N}\}$, where $\mathcal{K} := [K]$ is the index set of constraints, $d_{k,n}(a_n) \in \mathbb{R}_+$ represents the consumption of the k-th resource consumption by the n-th MDP when action a_n is taken, and $b_k \in \mathbb{R}_+$ the available resource of type k.[1] We define an idle action that consumes no resources for any resource k to ensure that the feasible action space is non-empty.

The state transitions of the sub-MDPs are independent, so the system transits from state $\boldsymbol{s}$ to state $\boldsymbol{s}'$ for a given feasible action $\boldsymbol{a}$ at time t with probability $p^{(N)}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) = \prod_{n=1}^{N} p_n(s'_n|s_n, a_n) = \prod_{n=1}^{N} \mathbb{P}(s_{t+1,n} = s'_n|s_{t,n} = s_n, a_{t,n} = a_n)$. After choosing an action $\boldsymbol{a} \in \mathcal{A}^{(N)}$ in the state $\boldsymbol{s} \in \mathcal{S}^{(N)}$, the decision maker receives rewards defined as $\boldsymbol{r}(\boldsymbol{s}, \boldsymbol{a}) = (r_1(s_1, a_1), \ldots, r_N(s_N, a_N))$ with each component representing the reward associated with the respective sub-MDP $\mathcal{M}_n$. We employ a vector form for the rewards to offer the flexibility for formulating fairness objectives on individual expected total discounted rewards in later sections.

We consider stationary Markovian policy $\pi : \mathcal{S}^{(N)} \times \mathcal{A}^{(N)} \to [0, 1]$, with notation $\pi(\boldsymbol{s}, \boldsymbol{a})$ capturing the probability of performing action $\boldsymbol{a}$ in state $\boldsymbol{s}$. The initial state $\boldsymbol{s}_0$ is sampled from the distribution $\mu^{(N)}$. Using the discounted-reward criteria, the state-value function V_n^π specific to the n-th sub-MDP $\mathcal{M}_n$, starting from an arbitrary initial state $\boldsymbol{s}_0$ under policy π, is defined as $V_n^\pi(\boldsymbol{s}_0) := \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_n(s_{t,n}, a_{t,n})|\boldsymbol{s}_0\right]$, where $\boldsymbol{a}_t \sim \pi(\boldsymbol{s}_t, \cdot)$. The joint state-value vector-valued function $\boldsymbol{V}^\pi(\boldsymbol{s}_0) : \mathcal{S}^{(N)} \to \mathbb{R}^N$ is defined as the column vector of expected total discounted rewards for all sub-MDPs under policy π, i.e., $\boldsymbol{V}^\pi(\boldsymbol{s}_0) := \left(V_1^\pi(\boldsymbol{s}_0), V_2^\pi(\boldsymbol{s}_0), \ldots, V_N^\pi(\boldsymbol{s}_0)\right)^\top$. We define $\boldsymbol{V}_0^\pi$ as the expected vectorial state-value under initial distribution $\mu^{(N)}$, i.e.,

$$\boldsymbol{V}_0^\pi := \mathbb{E}[\boldsymbol{V}^\pi(\boldsymbol{s}_0)|\boldsymbol{s}_0 \sim \mu^{(N)}]. \tag{1}$$

2.2 The Generalized Gini Function

The vector $\boldsymbol{V}_0^\pi$ represents expected utilities for users. A social welfare function aggregates these utilities into a scalar, measuring fairness in utility distribution with respect to a maximization objective.

Social welfare functions can vary depending on the values of a society, such as α-fairness (Mo and Walrand, 2000) or max-min fairness (Bistritz et al., 2020). Following Siddique et al. (2020), we require a fair solution to meet three properties: efficiency, impartiality, and equity. This motivates the use of the GGF from economics (Weymark, 1981), which satisfies these properties. For N users, the GGF is defined as $\mathrm{GGF}_{\boldsymbol{w}}[\boldsymbol{v}] := \min_{\sigma \in \mathbb{S}^N} \sum_{n=1}^{N} w_n v_{\sigma(n)}$, where $\boldsymbol{v} \in \mathbb{R}^N$, $\boldsymbol{w} \in \Delta(\mathcal{N})$ is non-increasing in n, i.e., $w_1 \geq w_2 \geq \cdots \geq w_N$. Intuitively, since $\mathrm{GGF}_{\boldsymbol{w}}[\boldsymbol{v}] = \sum_{n=1}^{N} w_n v_{\sigma^*(n)}$ with σ^* as the minimizer, which reorders the terms of $\boldsymbol{v}$ from lowest to largest, it computes the weighted sum of $\boldsymbol{v}$ assigning larger weights to its lowest components.

As discussed in Siddique et al. (2020), the GGF can reduce to special cases by setting the weights to specific values, including the maxmin egalitarian approach ($w_1 \to 1, w_2 \to 0, \ldots, w_N \to 0$) (Rawls, 1971), regularized maxmin egalitarian ($w_1 \to 1, w_2 \to \epsilon, \ldots, w_N \to \epsilon$), leximin notion of fairness ($w_k/w_{k+1} \to \infty$) (Rawls, 1971; Moulin, 1991), and the utilitarian approach formally defined below for the later use in reducing the GGF problem.

Definition 2.1 (Utilitarian Approach)**.** *The utilitarian approach within the GGF framework is obtained by setting equal weights for all individuals, i.e.,* $\boldsymbol{w}_{\mathbf{1}/N} := \mathbf{1}/N$ *so that* $\mathrm{GGF}_{\boldsymbol{w}_{\mathbf{1}/N}}[\boldsymbol{v}] = \min_{\sigma \in \mathbb{S}^N} \sum_{n=1}^{N} \frac{1}{N} v_{\sigma(n)} = \frac{1}{N} \sum_{n=1}^{N} v_n = \bar{v}$.

The utilitarian approach maximizes the average utilities over all individuals but does not guarantee fairness in utility distribution, as some users may be disadvantaged to increase overall utility. The GGF offers flexibility by encoding various fairness criteria in a structured way. Moreover, $\mathrm{GGF}_{\boldsymbol{w}}[\boldsymbol{v}]$ is concave in $\boldsymbol{v}$, which has nice properties for problem reformulation.

2.3 The GGF-WCMDP Problem

By combining the GGF and the vectored values from the WCMDP in (1), the goal of the GGF-WCMDP problem (2) is defined as finding a stationary policy π that maximizes the GGF of the expected total discounted rewards, i.e.,

[1] Actually, $b_k \leq \sum_{n=1}^{N} \max_{a_n \in \mathcal{A}_n} d_{k,n}(a_n)$, w.l.o.g.

$\max_{\pi} \mathrm{GGF}_{\boldsymbol{w}}\left[\boldsymbol{V}_0^{\pi}\right]$ that is equivalent to

$$\max_{\pi} \min_{\sigma \in \mathbb{S}_N} \boldsymbol{w}_{\sigma}^{\top} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{r}(\boldsymbol{s}_t, \boldsymbol{a}_t) \middle| \boldsymbol{s}_0 \sim \mu^{(N)}\right]. \tag{2}$$

We note that Lemma 3.1 in (Siddique et al., 2020) establishes the optimality of stationary Markov policies for any multi-objective discounted infinite-horizon MDP under the GGF criterion. The optimal policy for the GGF-WCMDP problem (2) can be computed by solving the following LP model with the GGF objective (GGF-LP):

$$\max_{\boldsymbol{\lambda}, \boldsymbol{\nu}, \boldsymbol{q}} \quad \sum_{i=1}^{N} \lambda_i + \sum_{j=1}^{N} \nu_j \tag{3a}$$

$$\text{s.t.} \quad \lambda_i + \nu_j \leq w_i \sum_{\boldsymbol{s} \in \mathcal{S}^{(N)}} \sum_{\boldsymbol{a} \in \mathcal{A}^{(N)}} \boldsymbol{r}_j(\boldsymbol{s}, \boldsymbol{a}) q(\boldsymbol{s}, \boldsymbol{a}), \quad \forall i, j \in \mathcal{N}, \tag{3b}$$

$$\sum_{\boldsymbol{a} \in \mathcal{A}^{(N)}} q(\boldsymbol{s}, \boldsymbol{a}) - \gamma \sum_{\boldsymbol{s}' \in \mathcal{S}^{(N)}} \sum_{\boldsymbol{a} \in \mathcal{A}^{(N)}} q(\boldsymbol{s}', \boldsymbol{a}) p^{(N)}(\boldsymbol{s} | \boldsymbol{s}', \boldsymbol{a}) = \mu^{(N)}(\boldsymbol{s}), \quad \forall \boldsymbol{s} \in \mathcal{S}^{(N)}, \tag{3c}$$

$$q(\boldsymbol{s}, \boldsymbol{a}) \geq 0, \quad \forall \boldsymbol{s} \in \mathcal{S}^{(N)}, \quad \forall \boldsymbol{a} \in \mathcal{A}^{(N)}. \tag{3d}$$

See Appendix D.1 for details on obtaining model (3) that exploits the dual linear programming formulation for solving discounted MDPs. Here, $q(\boldsymbol{s}, \boldsymbol{a})$ represents the total discounted visitation frequency for state-action pair $(\boldsymbol{s}, \boldsymbol{a})$, starting from $\boldsymbol{s}_0$.

The dual form separates dynamics from rewards, with the expected discounted reward for sub-MDP n given by $\sum_{\boldsymbol{s} \in \mathcal{S}^{(N)}} \sum_{\boldsymbol{a} \in \mathcal{A}^{(N)}} \boldsymbol{r}_n(\boldsymbol{s}, \boldsymbol{a}) q(\boldsymbol{s}, \boldsymbol{a})$. The one-to-one mapping between the solution $q(\boldsymbol{s}, \boldsymbol{a})$ and an optimal policy $\pi(\boldsymbol{s}, \boldsymbol{a})$ is $\pi(\boldsymbol{s}, \boldsymbol{a}) = q(\boldsymbol{s}, \boldsymbol{a}) / \sum_{\boldsymbol{a} \in \mathcal{A}^{(N)}} q(\boldsymbol{s}, \boldsymbol{a})$.

Scalability is a critical challenge in obtaining exact solutions as the state and action spaces grow exponentially with respect to the number of sub-MDPs, making the problem intractable. We thus explore approaches that exploit symmetric problem structures, apply count-based state aggregation, and use RL-based approximation methods, to address this scalability issue, which will be discussed next.

3 Utilitarian Reduction under Symmetric Sub-MDPs

In Section 3.1, we will formally define the concept of symmetric WCMDPs (definition 3.1) and prove that an optimal policy of the GGF-WCMDP problem can be obtained by solving the utilitarian WCMDP using "permutation invariant" policies. This enables the use of Whittle index policies in the RMAB setting while, for the more general setting, Section 3.2 proposes a count aggregation MDP reformulation that will be solved using deep RL in Section 4.

3.1 GGF-WCMDP Problem Reduction

We start with formally defining the conditions for a WCMDP to be considered symmetric.

Definition 3.1 (Symmetric WCMDP)**.** *A WCMDP is symmetric if*

1. ***(Identical Sub-MDPs)*** *Each sub-MDP is identical, i.e., $\mathcal{S}_n = \mathcal{S}$, $\mathcal{A}_n = \mathcal{A}$, $p_n = p$, $r_n = r$, $\mu_n = \mu$, for all $n \in \mathcal{N}$, and for some $(\mathcal{S}, \mathcal{A}, p, r, \mu, \gamma)$ tuple.*

2. ***(Symmetric resource consumption)*** *For any $k \in \mathcal{K}$, the number of resources consumed is the same for each sub-MDP, i.e., $d_{k,n}(a_n) = d_k(a_n)$ for all $n \in \mathcal{N}$, and for some $d_k(\cdot)$.*

3. ***(Permutation-Invariant Initial Distribution)*** *For any permutation operator $Q \in \mathcal{G}^N$, the probability of selecting the permuted initial state $Q\bar{\boldsymbol{s}}_0$ is equal to that of selecting $\bar{\boldsymbol{s}}_0$, i.e., $\mu^{(N)}(\bar{\boldsymbol{s}}_0) = \mu^{(N)}(Q\bar{\boldsymbol{s}}_0), \forall \bar{\boldsymbol{s}}_0 \in \mathcal{S}^{(N)}, \forall Q \in \mathcal{G}^N$.*

The conditions of symmetric WCMDP identify a class of WCMDPs that is invariant under any choice of indexing for the sub-MDPs. This gives rise to the notion of "permutation invariant" policies (see definition 1 in Cai et al. (2021)) and the question of whether this class of policies is optimal for symmetric WCMDPs.

Definition 3.2 (Permutation Invariant Policy)**.** *A Markov stationary policy π is said to be permutation invariant if the probability of selecting action $\boldsymbol{a}$ in state $\boldsymbol{s}$ is equal to that of selecting the permuted action $Q\boldsymbol{a}$ in the permuted state*

$Q\boldsymbol{s}$, for all $Q \in \mathcal{G}^N$. Formally, this can be expressed as $\pi(\boldsymbol{s}, \boldsymbol{a}) = \pi(Q\boldsymbol{s}, Q\boldsymbol{a})$, for all $Q \in \mathcal{G}^N$, $\boldsymbol{s} \in \mathcal{S}^{(N)}$ and $\boldsymbol{a} \in \mathcal{A}^{(N)}$.

This symmetry ensures that the expected state-value function, when averaged over all trajectories, is identical for each sub-MDP, leading to a uniform state-value representation. From this observation, and applying Theorem 6.9.1 from (Puterman, 2005), we construct a permutation-invariant policy from any policy, resulting in uniform state-value (Lemma 3.3).

Lemma 3.3 (Uniform State-Value Representation)**.** *If a WCMDP is symmetric, then for any policy π, there exists a corresponding permutation invariant policy $\bar{\pi}$ such that the vector of expected total discounted rewards for all sub-MDPs under $\bar{\pi}$ is equal to the average of the expected total discounted rewards for each sub-MDP, i.e., $\boldsymbol{V}_0^{\bar{\pi}} = \frac{1}{N}\sum_{n=1}^{N} V_{0,n}^{\pi}\mathbf{1}$.*

The proof is detailed in Appendix B.3. Furthermore, one can use the above lemma to show that the optimal policy for the GGF-WCMDP problem (2) under symmetry can be recovered from solving the problem with equal weights, i.e., the utilitarian approach. Our main result is presented in the following theorem. See Appendix B.4 for a detailed proof.

Theorem 3.4 (Utilitarian Reduction)**.** *For a symmetric WCMDP, let $\Pi^*_{\mathbf{1}/N,PI}$ be the set of optimal policies for the utilitarian approach that is permutation invariant, then $\Pi^*_{\mathbf{1}/N,PI}$ is necessarily non-empty and all $\pi^*_{\mathbf{1}/N,PI} \in \Pi^*_{\mathbf{1}/N,PI}$ satisfies*

$$\mathrm{GGF}_{\boldsymbol{w}}[\boldsymbol{V}_0^{\pi^*_{\mathbf{1}/N,PI}}] = \max_{\pi} \mathrm{GGF}_{\boldsymbol{w}}\left[\boldsymbol{V}_0^{\pi}\right], \forall \boldsymbol{w} \in \Delta(N).$$

This theorem simplifies solving the GGF-WCMDP problem by reducing it to an equivalent utilitarian problem, showing that at least one permutation-invariant policy is optimal for the original GGF-WCMDP problem and the utilitarian reduction. Therefore, we can restrict the search for optimal policies to this specific class of permutation-invariant policies. The utilitarian approach does not compromise the GGF optimality and allows us to leverage more efficient and scalable techniques to solve the GGF-WCMDP problem (Eq. 2), such as the Whittle index policies for RMABs, as demonstrated in the experimental section.

3.2 The Count Aggregation MDP

Assuming symmetry across all N sub-MDPs and using a permutation-invariant policy within a utilitarian framework allows us to simplify the global MDP by aggregating the sub-MDPs based on their state counts and tracking the number of actions taken in each state. Since each sub-MDP follows the same transition probabilities and reward structure, we can represent the entire system more compactly.

Motivated by the symmetry simplification representation in Gast et al. (2024) for the utilitarian objective, we consider an aggregation $\phi = (f, g_s)$, where $f : \mathcal{S}^{(N)} \to \mathbb{N}^S$ maps state $\boldsymbol{s}$ to a count representation $\boldsymbol{x}$ with x_s denoting the number of sub-MDPs in the s-th state. Similarly, $g_s : \mathcal{A}^{(N)} \to \mathbb{N}^{S\times A}$ maps action $\boldsymbol{a}$ to a count representation $\boldsymbol{u}$, where $u_{s,a}$ indicates the number of MDPs at s-th state that performs a-th action. We can formulate the count aggregation MDP (definition 3.5). The details on obtaining the exact form are in Appendix C.

Definition 3.5 (Count Aggregation MDP)**.** *The count aggregation MDP $\mathcal{M}_\phi$ derived from a WCMDP $(\mathcal{S}^{(N)}, \mathcal{A}^{(N)}, p^{(N)}, \boldsymbol{r}, \mu^{(N)}, \gamma)$ consists of the elements $(\mathcal{S}_f^{(N)}, \mathcal{A}_{g_s}^{(N)}, p_\phi^{(N)}, \bar{r}_\phi, \mu_f^{(N)}, \gamma)$.*

Both representations lead to the same optimization problem as established in Gast et al. (2024) when the objective is utilitarian. Using the count representation, the mean expected total discounted reward $\bar{V}_0^{\pi_{\mathbf{1}/N}}$ for a WCMDP $\mathcal{M}^{(N)}$ with permutation invariant distribution $\mu^{(N)}$ and equal weights $\boldsymbol{w}_{\mathbf{1}/N}$ (Theorem 3.4) is then equivalent to the expected total discounted mean reward $\bar{V}_0^{\pi_\phi}$ for the count aggregation MDP $\mathcal{M}_\phi$ given the policy $\pi_\phi : \mathcal{S}_f^{(N)} \to \Delta(\mathcal{A}_{g_s}^{(N)})$ under aggregation mapping with initial distribution $\mu_f^{(N)}$, i.e., $\bar{V}_0^{\pi_{\mathbf{1}/N}} = \frac{1}{N}\sum_{n=1}^{N} V_{0,n}^{\pi_{\mathbf{1}/N}} = \frac{1}{S}\sum_{s=1}^{S} V_{0,s}^{\pi_\phi} = \bar{V}_0^{\pi_\phi}$. The objective in Eq. 2 is therefore reformulated as $\max_{\pi_\phi} \bar{V}_0^{\pi_\phi}$, i.e.,

$$\max_{\pi_\phi} \frac{1}{S}\mathbb{E}_{\pi_\phi}\left[\sum_{t=0}^{\infty}\gamma^t \bar{r}_\phi(\boldsymbol{x}_t, \boldsymbol{u}_t)\middle| \boldsymbol{x}_0 \sim \mu_f^{(N)}\right]. \tag{4}$$

A LP method is provided to solve the count aggregation MDP in Appendix D.2.

4 Count-Proportion-Based DRL

We now consider the situation where the transition dynamics $p_\phi^{(N)}$ are unknown and the learner computes the (sub-)optimal policy through trial-and-error interactions with the environment. In Section 4.1, we introduce a count-proportion-based deep RL (CP-DRL) approach. This method incorporates a stochastic policy neural network with fixed-sized inputs and outputs, designed for optimizing resource allocation among stakeholders under constraints with count representation. In Section 4.2, we detail the priority based sampling procedure used to generate count actions.

4.1 Stochastic Policy Neural Network

One key property of the count aggregation MDP is that the dimensions of state space $\mathcal{S}_f^{(N)}$ and action space $\mathcal{A}_{g_s}^{(N)}$ are constant and irrespective of the number of sub-MDPs. To further simplify the analysis and eliminate the influence of N, we define the count state proportion as $\bar{\boldsymbol{x}} = \boldsymbol{x}/N$ and the resource proportion constraint for each resource k as $\bar{b}_k = b_k/(N \max_{a\in\mathcal{A}} d_k(a)) \in [0,1]$.

This converts the states into a probability distribution, allowing generalization when dealing with a large number of agents. The stochastic policy network in Figure 1 is designed to handle the reduced count aggregation MDP problem (4) by transforming the tuple $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{b}})$ into a priority score matrix $\boldsymbol{U}$ and a resource-to-use proportion vector $\tilde{\boldsymbol{p}}$, which are then used to generate count actions $\boldsymbol{u}$ via a sampling procedure (discussed in Section 4.2).

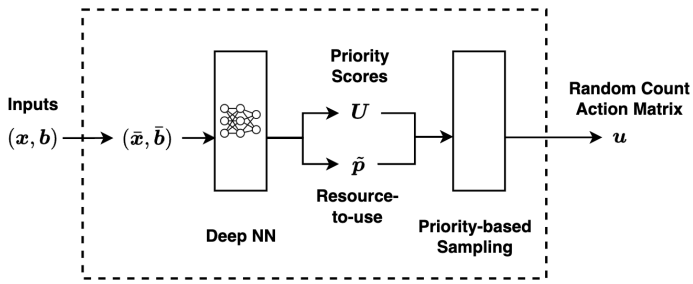


Figure 1: CP-based stochastic policy neural network.

The policy network features fixed-size inputs and outputs, enabling scalability in large-scale systems without requiring structural modifications when adjusting the number of resources or machines. The input consists of a fixed-size vector of size $S + K$, combining the count state proportion $\bar{\boldsymbol{x}} \in [0,1]^S$ and the resource proportion $\bar{\boldsymbol{b}} \in [0,1]^K$. The policy network processes these inputs to produce outputs of size $S \times A + K$, which include a matrix $\boldsymbol{U} \in (0,1]^{S\times A}$ representing the priority scores for selecting count actions and a vector $\tilde{\boldsymbol{p}} \in [0,1]^K$ representing the proportion of resource usage relative to the total available resources $\bar{\boldsymbol{b}}$.

The advantages of adding additional resource proportion nodes $\tilde{\boldsymbol{p}}$ to the output layer are twofold. First, it reduces the computational effort required to ensure that the resource-to-use does not exceed $\bar{\boldsymbol{b}}$. Instead, the resource constraint is satisfied by restricting the resource-to-use proportions $\tilde{\boldsymbol{p}}$ to be element-wise proportional to $\bar{\boldsymbol{b}}$. Second, since the optimal policy may not always use all available resources, we incorporate the additional nodes to capture the complex relationships between different states for more effective strategies to allocate resources.

4.2 Priority-based Sampling Procedure

Priority-based sampling presents a challenge since legal actions depend on state and resource constraints. To address this, a masking mechanism prevents the selection of invalid actions. Each element $u_{s,a}^p \in \boldsymbol{U}$ represents the priority score of taking the a-th action in the s-th state. When the state count x_s is zero, it implies the absence of sub-MDPs in this state, and the corresponding priority score is masked to zero. Legal priorities are thus defined for states with non-zero counts, i.e., $u_{s,a}^p = 0$ if $x_s = 0$ for all $s \in [S], a \in [A]$.

Since the selected state-action pairs must also satisfy multi-resource constraints, we introduced a forbidden set $\mathcal{F}$, which specifies the state-action pairs that are excluded from the sampling process. The complete procedure is outlined in Algorithm 1. The advantage of this approach is that the number of steps does not grow exponentially with the number of sub-MDPs. In the experiments, after obtaining the count action $\boldsymbol{u}$, a model simulator is used to generate

rewards and the next state as described in Algorithm 2 in Appendix E. The simulated outcomes are used for executing policy gradient updates and estimating state values.

Algorithm 1 Count Action Sampling Based on Priority Scores

Input: Count state $\boldsymbol{x}$, priority score matrix $\boldsymbol{U}$, resource limitations $\boldsymbol{b}$, resource-to-use proportion $\tilde{\boldsymbol{p}}$, resource consumption function $\boldsymbol{d}(a)$
Initialize: $\tilde{\boldsymbol{b}} \leftarrow \boldsymbol{b} \cdot \tilde{\boldsymbol{p}}$, $\boldsymbol{u} \leftarrow \mathbf{0}_{S \times A}$, $\mathcal{F} \leftarrow \emptyset$
Apply masking to $\boldsymbol{U}$ and update the forbidden set by $\mathcal{F} \leftarrow \mathcal{F} \cup \{(s,a) \mid u^p_{s,a} = 0 \text{ for } s \in [S] \text{ and } a \in [A]\}$.
while $|\mathcal{F}| < S \times A$ **do**
  Sample a state-action index pair $(s,a) \notin \mathcal{F}$ with the probability proportional to $\boldsymbol{U}$
  if $d_k(a) \leq \tilde{b}_k$ for all k **then**
    Update $u_{s,a} \leftarrow u_{s,a} + 1$, $x_s \leftarrow x_s - 1$
    Update $\tilde{b}_k \leftarrow \tilde{b}_k - d_k(a)$ for all k
    if $x_s = 0$ **then**
      Add all actions for the s-th state to forbidden set: $\mathcal{F} \leftarrow \mathcal{F} \cup \{(s,a) \mid \forall a \in [A]\}$
    end if
  else
    Add (s,a) to forbidden set $\mathcal{F} \leftarrow \mathcal{F} \cup \{(s,a)\}$.
  end if
end while
Return: Count action matrix $\boldsymbol{u}$

One critical advantage of using CP-DRL is its *scalability*. More specifically, the approach is designed to handle variable sizes of stakeholders N and resources K while preserving the number of aggregated count states constant for a given WCMDP. By normalizing inputs to fixed-size proportions, the network can seamlessly adapt to different scales, making it highly adaptable. Moreover, the fixed-size inputs allow *flexibility* that the neural network is trained once and used in multiple tasks with various numbers of stakeholders and resource limitations.

5 Experimental Results

We apply our methods to the machine replacement problem (Delage and Mannor, 2010; Akbarzadeh and Mahajan, 2019), providing a scalable framework for evaluating the CP-DRL approach as problem size and complexity increase. We focus on a single resource ($K = 1$) and binary action ($A = 2$) for each machine, allowing validation against the Whittle index policy for RMABs (Whittle, 1988). We applied various DRL algorithms, including *Soft Actor Critic* (SAC), *Advantage Actor Critic* (A2C), and *Proximal Policy Optimization* (PPO). Among these, PPO algorithm (Schulman et al., 2017) consistently delivers the most stable and high-quality performance. We thus choose PPO as the main algorithm for our CP-DRL approach (see Section 4). Our code is provided on Github[2].

Machine Replacement Problem The problem consists of N identical machines following Markovian deterioration rules with S states representing aging stages. The state space $\mathcal{S}^{(N)}$ is a Cartesian product. At each decision stage, actions $\boldsymbol{a}$ are applied to all machines under the resource constraint, with action a_n representing operation (*passive* action 0) or replace (*active* action 1). Resource consumption $d_n(a_n)$ is 1 for replacements and 0 for operation, with up to b replacements per time step. Costs range from 0 to 1, transformed to fit the reward representation by multiplying by -1 and adding 1. Machines degrade if not replaced and remain in state S until replaced. Refer to Appendix F.1 for cost structures and the transition probabilities. We choose the operational and replacement costs across two presets to capture different scenarios (see Appendix F.2 for details): *i) Exponential-RCCC* and *ii) Quadratic-RCCC*.

The goal is to find a fair policy that maximizes the GGF score over expected total discounted mean rewards with count aggregation MDP. In cases like electricity or telecommunication networks, where equipment is regionally distributed, a fair policy guarantees an equitable operation and replacement, thereby preventing frequent failures in specific areas that lead to unsatisfactory and unfair results for certain customers.

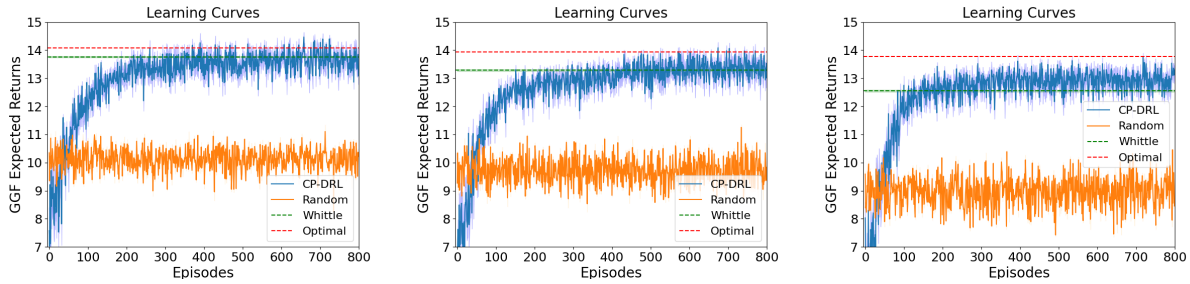
Experimental Setup We designed a series of experiments to test the GGF-optimality, flexibility, scalability, and efficiency of our CP-DRL algorithm. We compare against optimal solutions (OPT) from the GGF-LP model (3) for

[2] `https://anonymous.4open.science/r/GGF-WCMDP`

small instances solved with Gurobi 10.0.3, the Whittle index policy (WIP) for RMABs, and a random (RDM) agent that selects actions randomly at each time step and averages the results over 10 independent runs.

GGF weights decay exponentially with a factor of 2, defined as $\boldsymbol{w}_n = 1/2^n$, and normalized to sum to 1. We use a uniform distribution $\mu^{(N)}$ over $\mathcal{S}^{(N)}$ and set the discount factor $\gamma = 0.95$. We use Monte Carlo simulations to evaluate policies over M trajectories truncated at time length T. We choose M = 1,000 and T = 300 across all experiments. Hyperparameters for the CP-DRL algorithm are in Appendix F.3.

Experiment 1 (GGF-Optimality) We obtain optimal solutions using the OPT model for instances where $N \in \{3, 4, 5\}$, with each machine having $S = 3$ states. We select indexable instances to apply the WIP method for comparison. Note that, the WIP method is particularly effective in this case as it solves the equivalent utilitarian problem (as demonstrated in the utilitarian reduction result in Section 3.1). In most scenarios with small instances, WIP performs near-GGF-optimal since resources are assigned impartially, making it challenging for CP-DRL to consistently outperform WIP. As shown in Figure 2, the CP-DRL algorithm converges toward or slightly below the OPT values across the scenarios for the *Exponential-RCCC* case. WIP performs better than the random agent but does not reach the OPT values, especially as the number of machines increases. CP-DRL either outperforms or has an equivalent performance as WIP but consistantly outperforms the random policy.



(a) N=3 (b) N=4 (c) N=5

Figure 2: **(Colored) Learning curves for different numbers of machines (N from 3 to 5).** Experimental results for the Exponential-RCCC scenario are shown with y-axes starting at 7 for zoom-in. Red dashed lines represent the OPT values, green dashed lines show the WIP performance, blue lines depict CP-DRL learning curves over 800 episodes, and orange lines show the RDM performance. Shaded areas indicate the standard deviation across 5 runs.

Experiment 2 (Flexibility) The fixed-size input-output design allows CP-DRL to leverage multi-task training (MT) with varying machine numbers and resources. We refer to this multi-task extension as CP-DRL(MT). We trained the CP-DRL(MT) with $N \in \{2, 3, 4, 5\}$, randomly switching configurations at the end of each episode over 2000 training episodes. CP-DRL(MT) was evaluated separately, and GGF values for WIP and RDM policies were obtained from 1000 Monte Carlo runs. The numbers following the plus-minus sign ($\pm$) represent the variance across 5 experiments with different random seeds in Table 1 and 2. Variances for WIP and RDM are minimal and omitted, with bold font indicating the best GGF scores at each row excluding optimal values. As shown in Table 1, CP-DRL(MT) consistently achieves scores very close to the OPT values as the number of machines increases from 2 to 4. For the 5-machine case, CP-DRL(MT) shows slightly better performance than the single-task CP-DRL. In Table 2, the single- and multi-task CP-DRL agents show slight variations in performance across different machine numbers. For $N = 5$, CP-DRL achieves the best GGF score, slightly outperforming WIP.

Table 1: GGF Scores (Exponential-RCCC).

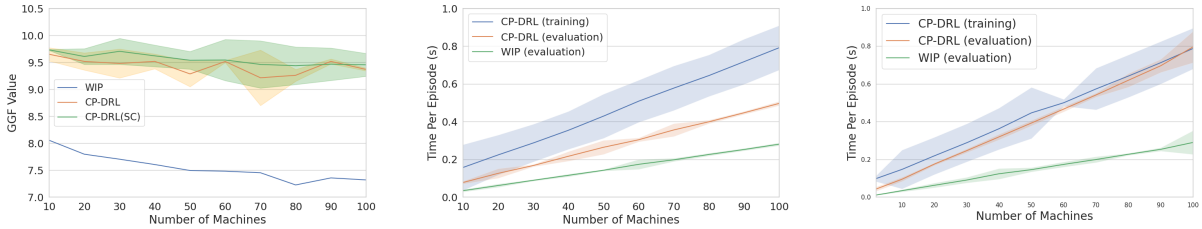
N	OPT	WIP	CP-DRL	CP-DRL(MT)	RDM
2	14.19	14.07	**14.12** $\pm$ 0.01	14.11 $\pm$ 0.01	9.67
3	14.08	13.75	**13.95** $\pm$ 0.02	13.89 $\pm$ 0.14	10.13
4	13.94	13.27	**13.64** $\pm$ 0.05	13.59 $\pm$ 0.10	9.74
5	13.77	12.47	12.96 $\pm$ 0.01	**13.28** $\pm$ 0.03	8.95

Table 2: GGF Scores (Quadratic-RCCC).

N	OPT	WIP	CP-DRL	CP-DRL(MT)	RDM
2	16.17	**16.17**	16.14 $\pm$ 0.00	16.14 $\pm$ 0.00	10.15
3	16.10	**16.09**	16.05 $\pm$ 0.00	16.05 $\pm$ 0.00	11.83
4	16.01	**16.01**	15.94 $\pm$ 0.00	15.94 $\pm$ 0.00	12.17
5	15.91	15.86	**15.87** $\pm$ 0.02	15.86 $\pm$ 0.02	11.98

Experiment 3 (Scalability) We assess CP-DRL scalability by increasing the number of machines while keeping the resource proportion at 0.1 for Exponential-RCCC instances. We refer to this scaled extension as CP-DRL(SC).

Machine numbers vary from 10 to 100 to evaluate CP-DRL performance as the problem size grows. We also use CP-DRL(SC), trained on 10 machines with 1 unit of resource, and scale it to tasks with 20 to 100 machines. Figure 3(a) shows CP-DRL and CP-DRL(SC) consistently achieve higher GGF values than WIP as machine numbers increase. CP-DRL(SC) delivers results comparable to separately trained CP-DRL, reducing training time while maintaining similar performance. Both WIP and CP-DRL show linear growth in time consumption per episode as machine numbers scale up.



(a) GGF values for the number of machines $N \in [10, 100]$.

(b) Time per episode in seconds with a resource ratio $b/N = 0.1$.

(c) Time per episode in seconds with a resource ratio $b/N = 0.5$.

Figure 3: **(Colored) Scalability and Time Efficiency of CP-DRL.** Subfigures (a) and (b) show the scalability of CP-DRL with a fixed resource ratio of 0.1. Subfigure (a) presents GGF values across different machine counts, with intervals representing standard deviation over 5 runs. Subfigure (b) and (c) depicts time per episode in seconds for a fixed resource ratio of 0.1 and 0.5, respectively. In all time plots, the green line represents WIP during MC evaluation, the blue line shows CP-DRL during training, and the orange line represents CP-DRL during MC evaluation.

Experiment 4 (Efficiency) In the GGF-LP model (3), the number of constraints grows exponentially with the number of machines N as $N^2 + S^N$, and the variables increase by $2N + (N+1) \cdot S^N$. Using the Count dual LP model (18) reduces the model size, but constraints still grow as $\binom{N+S-1}{S-1}$ and variables increase by $\binom{N+S-1}{S-1} \cdot A$. These growth patterns create computational challenges as problem size increases. Detailed time analysis for GGF-LP and Count dual LP models with N from 2 to 7 is provided in Appendix F.4. In addition to the time per episode for a fixed ratio of 0.1 in Figure 3a, we analyze performance with a 0.5 ratio (Figure 3c) and varying machine proportions, keeping the number of machines fixed at 10. We evaluate CP-DRL over machine proportions from 0.1 to 0.9. The results show that time per episode increases linearly with machine count, while the training and evaluation times remain relatively stable. This indicates that the sampling procedure for legal actions is the primary bottleneck. Meanwhile, the resource ratio has minimal impact on computing times.

6 Conclusion

We incorporate the fairness consideration in terms of the generalized Gini function within the weakly coupled Markov decision processes, and define the GGF-WCMDP optimization problem. First, we present an exact method based on linear programming for solving it. We then derive an equivalent problem based on a utilitarian reduction when the WCMDP is symmetric, and show that the set of optimal permutation invariant policy for the utilitarian objective is also optimal for the original GGF-WCMDP problem. We further leverage this result by utilizing a count state representation and introduce a count-proportion-based deep RL approach to devise more efficient and scalable solutions. Our empirical results show that the proposed method using PPO as the RL algorithm consistently achieves high-quality GGF solutions. Moreover, the flexibility provided by the count-proportion approach offers possibilities for scaling up to more complex tasks and context where Whittle index policies are unavailable due to the violation of the indexibility property by the sub-MDPs.

Acknowledgements

Xiaohui Tu was partially funded by GERAD and IVADO. Yossiri Adulyasak was partially supported by the Canadian Natural Sci- ences and Engineering Research Council [Grant RGPIN-2021-03264] and by the Canada Research Chair program [CRC-2022-00087]. Nima Akbarzadeh was partially funded by GERAD, FRQNT, and IVADO. Erick Delage

was partially supported by the Canadian Natural Sciences and Engineering Research Council [Grant RGPIN-2022-05261] and by the Canada Research Chair program [950-230057].

References

Adelman, D. and Mersereau, A. J. (2008). Relaxations of weakly coupled stochastic dynamic programs. *Operations Research*, 56(3):712–727.

Akbarzadeh, N. and Mahajan, A. (2019). Restless bandits with controlled restarts: Indexability and computation of whittle index. In *2019 IEEE 58th conference on decision and control (CDC)*, pages 7294–7300. IEEE.

Alamdari, P. A., Klassen, T. Q., Creager, E., and McIlraith, S. A. (2023). Remembering to be fair: On non-markovian fairness in sequential decisionmaking (preliminary report). *arXiv preprint arXiv:2312.04772*.

Atwood, J., Srinivasan, H., Halpern, Y., and Sculley, D. (2019). Fair treatment allocations in social networks. *arXiv preprint arXiv:1911.05489*.

Bistritz, I., Baharav, T., Leshem, A., and Bambos, N. (2020). My fair bandit: Distributed learning of max-min fairness with multi-player bandits. In *International Conference on Machine Learning*, pages 930–940. PMLR.

Boutilier, C. and Lu, T. (2016). Budget allocation using weakly coupled, constrained markov decision processes. In *UAI*.

Cai, D., Lim, S. H., and Wynter, L. (2021). Efficient reinforcement learning in resource allocation problems through permutation invariant multi-task learning. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 2270–2275.

Caton, S. and Haas, C. (2024). Fairness in machine learning: A survey. *ACM Computing Surveys*.

Chen, R. J., Wang, J. J., Williamson, D. F., Chen, T. Y., Lipkova, J., Lu, M. Y., Sahai, S., and Mahmood, F. (2023). Algorithmic fairness in artificial intelligence for medicine and healthcare. *Nature biomedical engineering*, 7(6):719–742.

Creager, E., Madras, D., Pitassi, T., and Zemel, R. (2020). Causal Modeling for Fairness In Dynamical Systems. In *Proceedings of the 37th International Conference on Machine Learning*, pages 2185–2195. PMLR.

D'Amour, A., Srinivasan, H., Atwood, J., Baljekar, P., Sculley, D., and Halpern, Y. (2020). Fairness is not static: Deeper understanding of long term fairness via simulation studies. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 525–534, Barcelona Spain. ACM.

Delage, E. and Mannor, S. (2010). Percentile Optimization for Markov Decision Processes with Parameter Uncertainty. *Operations Research*.

El Shar, I. and Jiang, D. (2024). Weakly coupled deep q-networks. *Advances in Neural Information Processing Systems*, 36.

Elmalaki, S. (2021). Fair-iot: Fairness-aware human-in-the-loop reinforcement learning for harnessing human variability in personalized iot. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, pages 119–132.

Farnadi, G., St-Arnaud, W., Babaki, B., and Carvalho, M. (2021). Individual fairness in kidney exchange programs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11496–11505.

Gajane, P., Saxena, A., Tavakol, M., Fletcher, G., and Pechenizkiy, M. (2022). Survey on fair reinforcement learning: Theory and practice. *arXiv preprint arXiv:2205.10032*.

Gast, N., Gaujal, B., and Yan, C. (2024). Reoptimization nearly solves weakly coupled markov decision processes.

Ge, Y., Zhao, X., Yu, L., Paul, S., Hu, D., Hsieh, C.-C., and Zhang, Y. (2022). Toward pareto efficient fairness-utility trade-off in recommendation through reinforcement learning. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 316–324.

Ghalme, G., Nair, V., Patil, V., and Zhou, Y. (2022a). Long-Term Resource Allocation Fairness in Average Markov Decision Process (AMDP) Environment. *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*.

Ghalme, G., Nair, V., Patil, V., and Zhou, Y. (2022b). Long-term resource allocation fairness in average markov decision process (amdp) environment. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 525–533.

Hassanzadeh, P., Kreacic, E., Zeng, S., Xiao, Y., and Ganesh, S. (2023). Sequential fair resource allocation under a markov decision process framework. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, pages 673–680.

Hawkins, J. T. (2003). *A Langrangian decomposition approach to weakly coupled dynamic optimization problems and its applications*. PhD thesis, Massachusetts Institute of Technology.

Herlihy, C., Prins, A., Srinivasan, A., and Dickerson, J. P. (2023). Planning to fairly allocate: Probabilistic fairness in the restless bandit setting. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 732–740.

Jabbari, S., Joseph, M., Kearns, M., Morgenstern, J., and Roth, A. (2017). Fairness in reinforcement learning. In *International conference on machine learning*, pages 1617–1626. PMLR.

Jiang, J. and Lu, Z. (2019). Learning fairness in multi-agent systems. *Advances in Neural Information Processing Systems*, 32.

Kozodoi, N., Jacob, J., and Lessmann, S. (2022). Fairness in credit scoring: Assessment, implementation and profit implications. *European Journal of Operational Research*, 297(3):1083–1094.

Li, D. and Varakantham, P. (2022a). Efficient resource allocation with fairness constraints in restless multi-armed bandits. In *Uncertainty in Artificial Intelligence*, pages 1158–1167. PMLR.

Li, D. and Varakantham, P. (2022b). Towards soft fairness in restless multi-armed bandits. *arXiv preprint arXiv:2207.13343*.

Li, D. and Varakantham, P. (2023). Avoiding starvation of arms in restless multi-armed bandit. *International Foundation for Autonomous Agents and Multiagent Systems*.

Liu, C., Chen, C.-X., and Chen, C. (2021). Meta: A city-wide taxi repositioning framework based on multi-agent reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):13890–13895.

Liu, L. T., Dean, S., Rolf, E., Simchowitz, M., and Hardt, M. (2018). Delayed Impact of Fair Machine Learning. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3150–3158. PMLR.

Liu, W., Liu, F., Tang, R., Liao, B., Chen, G., and Heng, P. A. (2020). Balancing between accuracy and fairness for interactive recommendation with reinforcement learning. In *Pacific-asia conference on knowledge discovery and data mining*, pages 155–167. Springer.

Mo, J. and Walrand, J. (2000). Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on networking*, 8(5):556–567.

Moulin, H. (1991). *Axioms of cooperative decision making*. Cambridge university press.

Puranik, B., Madhow, U., and Pedarsani, R. (2022). Dynamic positive reinforcement for long-term fairness. In *ICML 2022 Workshop on Responsible Decision Making in Dynamic Environments*.

Puterman, M. L. (2005). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Raghavan, M., Barocas, S., Kleinberg, J., and Levy, K. (2020). Mitigating bias in algorithmic hiring: Evaluating claims and practices. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 469–481.

Rawls, J. (1971). A theory of justice. *Cambridge (Mass.)*.

Reuel, A. and Ma, D. (2024). Fairness in reinforcement learning: A survey.

Saure, A., Patrick, J., Tyldesley, S., and Puterman, M. L. (2012). Dynamic multi-appointment patient scheduling for radiation therapy. *European Journal of Operational Research*, 223(2):573–584.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Segal, M., George, A.-M., and Dimitrakakis, C. (2023). Policy fairness and unknown bias dynamics in sequential allocations. In *Proceedings of the 3rd ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization*, EAAMO '23, New York, NY, USA. Association for Computing Machinery.

Siddique, U., Sinha, A., and Cao, Y. (2023). Fairness in preference-based reinforcement learning. *arXiv preprint arXiv:2306.09995*.

Siddique, U., Weng, P., and Zimmer, M. (2020). Learning fair policies in multi-objective (deep) reinforcement learning with average and discounted rewards. In *International Conference on Machine Learning*, pages 8905–8915. PMLR.

Sood, A., Jain, S., and Gujar, S. (2024). Fairness of exposure in online restless multi-armed bandits. *arXiv preprint arXiv:2402.06348*.

van den Broek, E., Sergeeva, A., and Huysman, M. (2020). Hiring algorithms: An ethnography of fairness in practice. In *40th international conference on information systems, ICIS 2019*, pages 1–9. Association for Information Systems.

Verma, S., Zhao, Y., Shah, S., Boehmer, N., Taneja, A., and Tambe, M. (2024). Group fairness in predict-then-optimize settings for restless bandits. In *The 40th Conference on Uncertainty in Artificial Intelligence*.

Wen, M., Bastani, O., and Topcu, U. (2021). Algorithms for fairness in sequential decision making. In *International Conference on Artificial Intelligence and Statistics*, pages 1144–1152. PMLR.

Weymark, J. A. (1981). Generalized gini inequality indices. *Mathematical Social Sciences*, 1(4):409–430.

Whittle, P. (1988). Restless bandits: Activity allocation in a changing world. *Journal of applied probability*, 25(A):287–298.

Yu, G., Siddique, U., and Weng, P. (2023). Fair deep reinforcement learning with preferential treatment. In *ECAI*, pages 2922–2929.

Zhang, X. (2022). *Near-Optimality for Multi-action Multi-resource Restless Bandits with Many Arms*. PhD thesis, Cornell University.

Zhang, X., Khaliligarekani, M., Tekin, C., et al. (2019). Group retention when using machine learning in sequential decision making: the interplay between user dynamics and fairness. *Advances in Neural Information Processing Systems*, 32.

Zhang, X., Tu, R., Liu, Y., Liu, M., Kjellstrom, H., Zhang, K., and Zhang, C. (2020). How do fair decisions fare in long-term qualification? *Advances in Neural Information Processing Systems*, 33:18457–18469.

Zhao, H. and Gordon, G. (2019). Inherent tradeoffs in learning fair representations. *Advances in neural information processing systems*, 32.

Zimmer, M., Glanois, C., Siddique, U., and Weng, P. (2021). Learning fair policies in decentralized cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 12967–12978. PMLR.

A Related Work

Fairness-aware learning is increasingly integrated into the decision-making ecosystem to accommodate minority interests. However, naively imposing fairness constraints can actually exacerbate inequity (Wen et al., 2021) if the feedback effects of the decisions are ignored. Many real-world fairness applications are not one-time static decisions (Zhao and Gordon, 2019) and can thus be better modeled with sequential decision problems, which still remain relatively under-studied.

Fairness with dynamics There are a few studies investigating fairness-aware sequential decision making. For instance, Liu et al. (2018) consider one-step delayed feedback effects, Creager et al. (2020) propose causal modeling of dynamical systems to address fairness, and Zhang et al. (2019) construct a user participation dynamics model where individuals respond to perceived decisions by leaving the system uniformly at random. These studies extend the fairness definition in temporally extended decision-making settings, but do not take feedback and learning into account that the system may fail to adapt to changing conditions. Alamdari et al. (2023) address this gap by introducing multi-stakeholder fairness as non-Markovian sequential decision-making and developing a Q-learning based algorithm with counterfactual experiences to enhance sample-efficient fair policy learning.

Fairness in Markov decision processes Zhang et al. (2020) consider how algorithmic decisions impact the evolution of feature space of the underlying population modeled as Markov decision processes (MDP) but is limited to binary decisions. Ghalme et al. (2022b) study a fair resource allocation problem in the average MDP setting and proposes an approximate algorithm to compute the policy with sample complexity bounds. However, their definition of fairness is restricted to the minimum visitation frequency across all states, potentially resulting in an unbalanced rewards among users. Wen et al. (2021) develop fair decision-making policies in discounted MDPs, but the performance guarantees are achieved only under a loose condition. In contrast, our work takes into account a more comprehensive definition of fairness. Segal et al. (2023) investigate the impact of societal bias dynamics on long-term fairness and the interplay between utility and fairness under various optimization parameters. Additionally, Hassanzadeh et al. (2023) address a fair resource allocation problem similar to our work but in continuous state and action space. They define fairness to the agents considering all their allocations over the horizon under the Nash Social Welfare objective in hindsight.

Fairness in reinforcement learning Jabbari et al. (2017) initiate the *meritocratic fairness* notion from the multi-arm bandits setting to the reinforcement learning (RL) setting. Later, fairness consideration has been integrated in reinforcement learning to achieve fair solutions in different domains, including a fair vaccine allocation policy that equalizes outcomes in the population (Atwood et al., 2019), balancing between fairness and accuracy for interactive user recommendation (Liu et al., 2020; Ge et al., 2022), and fair IoT that continuously monitors the human state and changes in the environment to adapt its behavior accordingly (Elmalaki, 2021). However, most work focuses on the impartiality aspect of fairness. Jiang and Lu (2019) investigate multi-agent RL where fairness is defined over agents and encoded with a different welfare function, but the focus is on learning decentralized policies in a distributed way. We refer readers to two literature review papers by Gajane et al. (2022) and Reuel and Ma (2024) on fairness considerations in RL, which provide comprehensive insights into current trends, challenges, and methodologies in the field.

Fairness in restless multi-arm bandits A line of work closely related to ours focuses on fairness in restless multi-arm bandits (RMABs). Li and Varakantham (2022a) first introduce the consideration of fairness in restless bandits by proposing an algorithm that ensures a minimum number of selections for each arm. Subsequent studies have explored similar individual fairness constraints, which aim to distribute resources equitably among arms but in a probabilistic manner. For instance, Herlihy et al. (2023) introduce a method that imposes a strictly positive lower bound on the probability of each arm being pulled at each timestep. Li and Varakantham (2022b, 2023) investigate fairness by always probabilistically favoring arms that yield higher long-term cumulative rewards. Additionally, Sood et al. (2024) propose an approach where each arm receives pulls in proportion to its merit, which is determined by its stationary reward distribution. Our work differs from these approaches by explicitly aiming to prevent disparity and ensure a more balanced reward distribution among all arms through the generalized Gini welfare objective. The only work that considers the Gini index objective is by Verma et al. (2024), which develops a decision-focused learning pipeline to solve equitable RMABs. In contrast, our work applies to a more general setting on weakly coupled MDPs, and does not rely on the Whittle indexability of the coupled MDPs.

B Proofs of Section 3

We start this section with some preliminary results regarding 1) the effect of replacing a policy with one with permuted indices on the value function of a symmetric WCMDP (Section B.1); and 2) a well known result due to Puterman (2005) on the equivalency between stationary policies and occupancy measures (Section B.2). This is followed with the proof of Lemma 3.3 and our main Theorem 3.4 in sections B.3 and B.4 respectively.

B.1 Value Function under Permuted Policy for Symmetric WCMDP

Lemma B.1. *If a weakly coupled WCMDP is symmetric (definition 3.1), then for any policy π and permutation operator Q, we have $\boldsymbol{V}_0^{\pi} = Q\boldsymbol{V}_0^{\pi^Q}$, where the permuted policy $\pi^Q(\boldsymbol{s}, \boldsymbol{a}) := \pi(Q\boldsymbol{s}, Q\boldsymbol{a})$ for all $(\boldsymbol{s}, \boldsymbol{a})$ pairs.*

This lemma implies an important equivalency in symmetric weakly coupled MDPs with identical sub-MDPs. If we permute the states and actions of a policy, the permuted version of the resulting value function is equivalent to the original value function.

Proof: We can first show that for all t,

$$\mathbb{P}^{\pi^Q}(\boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a}|\boldsymbol{s}_0 = \bar{\boldsymbol{s}}_0) = \mathbb{P}^{\pi}(\boldsymbol{s}_t = Q\boldsymbol{s}, \boldsymbol{a}_t = Q\boldsymbol{a}|\boldsymbol{s}_0 = Q\bar{\boldsymbol{s}}_0).$$

This can be done inductively. Starting at $t = 0$, we have that:

$$\begin{aligned}\mathbb{P}^{\pi^Q}(\boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}|\boldsymbol{s}_0 = \bar{\boldsymbol{s}}_0) &= \pi^Q(\boldsymbol{s}, \boldsymbol{a})\mathbb{I}\{\boldsymbol{s} = \bar{\boldsymbol{s}}_0\} \\ &= \pi(Q\boldsymbol{s}, Q\boldsymbol{a})\mathbb{I}\{Q\boldsymbol{s} = Q\bar{\boldsymbol{s}}_0\} \\ &= \mathbb{P}^{\pi}(\boldsymbol{s}_0 = Q\boldsymbol{s}, \boldsymbol{a}_0 = Q\boldsymbol{a}|\boldsymbol{s}_0 = Q\bar{\boldsymbol{s}}_0).\end{aligned}$$

Next, assuming that $\mathbb{P}^{\pi^Q}(\boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a}|\boldsymbol{s}_0 = \bar{\boldsymbol{s}}_0) = \mathbb{P}^{\pi}(\boldsymbol{s}_t = Q\boldsymbol{s}, \boldsymbol{a}_t = Q\boldsymbol{a}|\boldsymbol{s}_0 = Q\bar{\boldsymbol{s}}_0)$, we can show that it is also the case for $t + 1$:

$$\begin{aligned}&\mathbb{P}^{\pi^Q}(\boldsymbol{s}_{t+1} = \boldsymbol{s}', \boldsymbol{a}_{t+1} = \boldsymbol{a}'|\boldsymbol{s}_0 = \bar{\boldsymbol{s}}_0) \\ &= \pi^Q(\boldsymbol{s}', \boldsymbol{a}')\sum_{\boldsymbol{s},\boldsymbol{a}}\mathbb{P}(\boldsymbol{s}_{t+1} = \boldsymbol{s}'|\boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a})\mathbb{P}^{\pi^Q}(\boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a}|\boldsymbol{s}_0 = \bar{\boldsymbol{s}}_0) \\ &= \pi(Q\boldsymbol{s}', Q\boldsymbol{a}')\sum_{\boldsymbol{s},\boldsymbol{a}}p^{(N)}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})\mathbb{P}^{\pi}(\boldsymbol{s}_t = Q\boldsymbol{s}, \boldsymbol{a}_t = Q\boldsymbol{a}|\boldsymbol{s}_0 = Q\bar{\boldsymbol{s}}_0) \\ &= \pi(Q\boldsymbol{s}', Q\boldsymbol{a}')\sum_{\boldsymbol{s},\boldsymbol{a}}p^{(N)}(Q\boldsymbol{s}'|Q\boldsymbol{s}, Q\boldsymbol{a})\mathbb{P}^{\pi}(\boldsymbol{s}_t = Q\boldsymbol{s}, \boldsymbol{a}_t = Q\boldsymbol{a}|\boldsymbol{s}_0 = Q\bar{\boldsymbol{s}}_0) \\ &= \mathbb{P}^{\pi}(\boldsymbol{s}_{t+1} = Q\boldsymbol{s}', \boldsymbol{a}_{t+1} = Q\boldsymbol{a}'|\boldsymbol{s}_0 = Q\bar{\boldsymbol{s}}_0),\end{aligned}$$

where we used the fact that the sub-MDPs are identical so that $p^{(N)}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) = p^{(N)}(Q\boldsymbol{s}'|Q\boldsymbol{s}, Q\boldsymbol{a})$.

We now have that,

$$\begin{aligned}\boldsymbol{V}_0^{\pi^Q} &= \sum_{\boldsymbol{s},\boldsymbol{a}}\sum_{\bar{\boldsymbol{s}}_0}\mu^{(N)}(\bar{\boldsymbol{s}}_0)\sum_{t=0}^{\infty}\gamma^t\mathbb{P}^{\pi^Q}(\boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a}|\boldsymbol{s}_0 = \bar{\boldsymbol{s}}_0)\boldsymbol{r}(\boldsymbol{s}, \boldsymbol{a}) \\ &= \sum_{\boldsymbol{s},\boldsymbol{a}}\sum_{\bar{\boldsymbol{s}}_0}\mu^{(N)}(\bar{\boldsymbol{s}}_0)\sum_{t=0}^{\infty}\gamma^t\mathbb{P}^{\pi}(\boldsymbol{s}_t = Q\boldsymbol{s}, \boldsymbol{a}_t = Q\boldsymbol{a}|\boldsymbol{s}_0 = Q\bar{\boldsymbol{s}}_0)\boldsymbol{r}(\boldsymbol{s}, \boldsymbol{a}) \\ &= \sum_{\boldsymbol{s},\boldsymbol{a}}\sum_{\bar{\boldsymbol{s}}_0}\mu^{(N)}(Q\bar{\boldsymbol{s}}_0)\sum_{t=0}^{\infty}\gamma^t\mathbb{P}^{\pi}(\boldsymbol{s}_t = Q\boldsymbol{s}, \boldsymbol{a}_t = Qa|\boldsymbol{s}_0 = Q\bar{\boldsymbol{s}}_0)\boldsymbol{r}(\boldsymbol{s}, \boldsymbol{a}) \\ &= \sum_{\boldsymbol{s},\boldsymbol{a}}\sum_{\bar{\boldsymbol{s}}_0}\mu^{(N)}(Q\bar{\boldsymbol{s}}_0)\sum_{t=0}^{\infty}\gamma^t\mathbb{P}^{\pi}(\boldsymbol{s}_t = Q\boldsymbol{s}, \boldsymbol{a}_t = Q\boldsymbol{a}|\boldsymbol{s}_0 = Q\bar{\boldsymbol{s}}_0)Q^{-1}\boldsymbol{r}(Q\boldsymbol{s}, Q\boldsymbol{a}) \\ &= Q^{-1}\left(\sum_{\boldsymbol{s},\boldsymbol{a}}\sum_{\bar{\boldsymbol{s}}_0}\mu^{(N)}(Q\bar{\boldsymbol{s}}_0)\sum_{t=0}^{\infty}\gamma^t\mathbb{P}^{\pi}(\boldsymbol{s}_t = Q\boldsymbol{s}, \boldsymbol{a}_t = Q\boldsymbol{a}|\boldsymbol{s}_0 = Q\bar{\boldsymbol{s}}_0)\boldsymbol{r}(Q\boldsymbol{s}, Q\boldsymbol{a})\right) \\ &= Q^{-1}\left(\sum_{\boldsymbol{s}',\boldsymbol{a}'}\sum_{\bar{\boldsymbol{s}}'_0}\mu^{(N)}(\bar{\boldsymbol{s}}'_0)\sum_{t=0}^{\infty}\gamma^t\mathbb{P}^{\pi}(\boldsymbol{s}_t = \boldsymbol{s}', \boldsymbol{a}_t = \boldsymbol{a}'|\boldsymbol{s}_0 = \bar{\boldsymbol{s}}'_0)\boldsymbol{r}(\boldsymbol{s}', \boldsymbol{a}')\right) \;=\; Q^{-1}\boldsymbol{V}_0^{\pi},\end{aligned}$$

where we first use the relation between P^{π^Q} and π, then exploit the permutation invariance of $\mu^{(N)}$. We then exploit the permutation invariance $Q\boldsymbol{r}(\boldsymbol{s},\boldsymbol{a}) = \boldsymbol{r}(Q\boldsymbol{s}, Q\boldsymbol{a})$, and reindex the summations using $\boldsymbol{s}' := Q\boldsymbol{s}$, $\boldsymbol{a}' := Q\boldsymbol{a}$, and $\bar{\boldsymbol{s}}_0' := Q\bar{\boldsymbol{s}}_0$. □

B.2 Mapping between stationary policies and occupancy measures

We present results of Theorem 6.9.1 in Puterman (2005) to support Lemma 3.3. A detailed proof is provided in the book.

Lemma B.2. *(Theorem 6.9.1 of Puterman (2005)) Let Π denote the set of stationary stochastic Markov policies and $\mathcal{X}$ the set of occupancy measures. There exists a bijection $h : \Pi \to \mathcal{X}$ such that for any policy π, $h(\pi)$ uniquely corresponds to its occupancy measure q_π. Specifically, there is a one-to-one mapping between policies and occupancy measures satisfying:*

1. *For any policy $\pi \in \Pi$, the occupancy measure $q_\pi : \mathcal{S}^{(N)} \times \mathcal{A}^{(N)} \to \mathbb{R}$ is defined as*

$$q_\pi(\boldsymbol{s},\boldsymbol{a}) := \sum_{\bar{\boldsymbol{s}}_0 \in \mathcal{S}^{(N)}} \mu^{(N)}(\bar{\boldsymbol{s}}_0) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}^\pi\left(\boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a} | \boldsymbol{s}_0 = \bar{\boldsymbol{s}}_0\right), \tag{5}$$

   for all $\boldsymbol{a} \in \mathcal{A}^{(N)}$ and $\boldsymbol{s} \in \mathcal{S}^{(N)}$.

2. *For any occupancy measure $q(\boldsymbol{s},\boldsymbol{a}) : \mathcal{S}^{(N)} \times \mathcal{A}^{(N)} \to \mathbb{R}$, the policy π_q is constructed as*

$$\pi_q(\boldsymbol{s},\boldsymbol{a}) := \frac{q(\boldsymbol{s},\boldsymbol{a})}{\sum\limits_{\boldsymbol{a}' \in \mathcal{A}^{(N)}} q\left(\boldsymbol{s},\boldsymbol{a}'\right)}, \tag{6}$$

   for all $\boldsymbol{a} \in \mathcal{A}^{(N)}$ and $\boldsymbol{s} \in \mathcal{S}^{(N)}$.

It follows that $\pi = \pi_{q_\pi}$.

Now, we show that the value function can be represented using occupancy measures.

Lemma B.3. *For any policy $\pi \in \Pi$, and the occupancy measure q_π defined by (5), the expected total discounted rewards under the policy π can be expressed as:*

$$\boldsymbol{V}_0^\pi = \sum_{\boldsymbol{s} \in \mathcal{S}^{(N)}} \sum_{\boldsymbol{a} \in \mathcal{A}^{(N)}} q_\pi(\boldsymbol{s},\boldsymbol{a}) \boldsymbol{r}(\boldsymbol{s},\boldsymbol{a}). \tag{7}$$

Proof. Expanding the expected total discounted rewards $\boldsymbol{V}_0^\pi$ (as defined by Equation 1), we have:

$$\boldsymbol{V}_0^\pi = \sum_{\bar{\boldsymbol{s}}_0 \in \mathcal{S}^{(N)}} \mu^{(N)}(\bar{\boldsymbol{s}}_0) \sum_{\boldsymbol{s} \in \mathcal{S}^{(N)}} \sum_{\boldsymbol{a} \in \mathcal{A}^{(N)}} \sum_{t=0}^{\infty} \mathbb{P}^\pi\left(\boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a} | \boldsymbol{s}_0 = \bar{\boldsymbol{s}}_0\right) \gamma^t \boldsymbol{r}(\boldsymbol{s},\boldsymbol{a}).$$

Rearranging the terms:

$$\boldsymbol{V}_0^\pi = \sum_{\boldsymbol{s} \in \mathcal{S}^{(N)}} \sum_{\boldsymbol{a} \in \mathcal{A}^{(N)}} \left(\sum_{\bar{\boldsymbol{s}}_0 \in \mathcal{S}^{(N)}} \mu^{(N)}(\bar{\boldsymbol{s}}_0) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}^\pi\left(\boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a} | \boldsymbol{s}_0 = \bar{\boldsymbol{s}}_0\right) \right) \boldsymbol{r}(\boldsymbol{s},\boldsymbol{a}).$$

By replacing the term in parentheses as the occupancy measure $q_\pi(\boldsymbol{s},\boldsymbol{a})$ in equation 5:

$$\boldsymbol{V}_0^\pi = \sum_{\boldsymbol{s} \in \mathcal{S}^{(N)}} \sum_{\boldsymbol{a} \in \mathcal{A}^{(N)}} q_\pi(\boldsymbol{s},\boldsymbol{a}) \boldsymbol{r}(\boldsymbol{s},\boldsymbol{a}).$$

This completes the proof. □

B.3 Proof of Lemma 3.3

Lemma 3.3 (Uniform State-Value Representation) *If a WCMDP is symmetric (definition 3.1), then for any policy π, there exists a corresponding permutation invariant policy $\bar{\pi}$ such that the vector of expected total discounted rewards for all sub-MDPs under $\bar{\pi}$ is equal to the average of the expected total discounted rewards for each sub-MDP, i.e.,*

$$\boldsymbol{V}_0^{\bar{\pi}} = \frac{1}{N} \sum_{n=1}^{N} V_{0,n}^\pi \mathbf{1}.$$

Proof by construction. We first construct, for any fixed Q, the permuted policy $\pi^Q(\boldsymbol{s}, \boldsymbol{a}) := \pi(Q\boldsymbol{s}, Q\boldsymbol{a})$ and characterize its occupancy measure q_{π^Q} as

$$q_{\pi^Q}(\boldsymbol{s}, \boldsymbol{a}) := \sum_{\bar{\boldsymbol{s}}_0 \in \mathcal{S}^{(N)}} \mu^{(N)}(\bar{\boldsymbol{s}}_0) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}^{\pi^Q}\left(\boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a} | \boldsymbol{s}_0 = \bar{\boldsymbol{s}}_0\right). \tag{8}$$

Next, we construct a new measure $\bar{q}$ obtained by averaging all permuted occupancy measures q_{π^Q} for $Q \in \mathcal{G}^N$ on all $(\boldsymbol{s}, \boldsymbol{a})$ pairs as

$$\bar{q}(\boldsymbol{s}, \boldsymbol{a}) := \frac{1}{N!} \sum_{Q} q_{\pi^Q}(\boldsymbol{s}, \boldsymbol{a}). \tag{9}$$

One can confirm that $\bar{q}$ is an occupancy measure, i.e., $\bar{q} \in \mathcal{X}$, since each $q_\pi^Q \in \mathcal{X}$ and $\mathcal{X}$ is convex. Indeed, the convexity of $\mathcal{X}$ easily follows from that the fact that it contains any measure that it is the set of measures that satisfy constraints 3c and 3d.

From Lemma B.2, a stationary policy $\bar{\pi}$ can be constructed such that its occupancy measure matches $\bar{q}(\boldsymbol{s}, \boldsymbol{a})$. Namely,

$$q_{\bar{\pi}}(\boldsymbol{s}, \boldsymbol{a}) := \sum_{\bar{\boldsymbol{s}}_0 \in \mathcal{S}^{(N)}} \mu(\bar{\boldsymbol{s}}_0) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}^{\bar{\pi}}\left(\boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a} | \boldsymbol{s}_0 = \bar{\boldsymbol{s}}_0\right) = \bar{q}(\boldsymbol{s}, \boldsymbol{a}), \forall \boldsymbol{s}, \boldsymbol{a}.$$

We can then derive the following steps:

$$\begin{aligned}
\boldsymbol{V}_0^{\bar{\pi}} &= \sum_{\boldsymbol{s} \in \mathcal{S}^{(N)}} \sum_{\boldsymbol{a} \in \mathcal{A}^{(N)}} q_{\bar{\pi}}(\boldsymbol{s}, \boldsymbol{a}) \boldsymbol{r}(\boldsymbol{s}, \boldsymbol{a}) \quad \text{(By Lemma B.3)} \\
&= \sum_{\boldsymbol{s} \in \mathcal{S}^{(N)}} \sum_{\boldsymbol{a} \in \mathcal{A}^{(N)}} \bar{q}(\boldsymbol{s}, \boldsymbol{a}) \boldsymbol{r}(\boldsymbol{s}, \boldsymbol{a}) \quad \text{(By Lemma B.2)} \\
&= \frac{1}{N!} \sum_{Q \in \mathcal{G}^N} \sum_{\boldsymbol{s} \in \mathcal{S}^{(N)}} \sum_{\boldsymbol{a} \in \mathcal{A}^{(N)}} q_{\pi^Q}(\boldsymbol{s}, \boldsymbol{a}) \boldsymbol{r}(\boldsymbol{s}, \boldsymbol{a}) \quad \text{(By construction in equation 9)} \\
&= \frac{1}{N!} \sum_{Q \in \mathcal{G}^N} \boldsymbol{V}_0^{\pi^Q} \quad \text{(By Lemma B.3)} \\
&= \frac{1}{N!} \sum_{Q \in \mathcal{G}^N} Q^{-1} \boldsymbol{V}_0^{\pi} \quad \text{(By Lemma B.1)} \\
&= \frac{1}{N!} \sum_{Q \in \mathcal{G}^N} Q^{-1} \begin{bmatrix} V_{0,1}^{\pi} \\ \cdots \\ V_{0,N}^{\pi} \end{bmatrix} \quad \text{(Vector form)} \\
&= \frac{1}{N!} \begin{bmatrix} (N-1)! \sum_{n=1}^{N} V_{0,n}^{\pi} \\ \cdots \\ (N-1)! \sum_{n=1}^{N} V_{0,n}^{\pi} \end{bmatrix} \quad \text{(Property of permutation group)} \\
&= \frac{1}{N!} (N-1)! \sum_{n=1}^{N} V_{0,n}^{\pi} \mathbf{1} \\
&= \frac{1}{N} \sum_{n=1}^{N} V_{0,n}^{\pi} \mathbf{1}.
\end{aligned}$$

We complete this proof by demonstrating that $\bar{\pi}$ is permutation invariant. Namely, for all $Q \in \mathcal{G}^N$, we can show

that:

$$\begin{aligned}\bar{\pi}(Q\boldsymbol{s}, Q\boldsymbol{a}) &\propto \frac{1}{N!}\sum_{Q'\in\mathcal{G}^N} q_{\pi^{Q'}}(Q\boldsymbol{s}, Q\boldsymbol{a}) \\ &= \frac{1}{N!}\sum_{Q'\in\mathcal{G}^N} q_{\pi}(Q'Q\boldsymbol{s}, Q'Q\boldsymbol{a}) \\ &= \frac{1}{N!}\sum_{Q''\in\mathcal{G}^N} q_{\pi}(Q''\boldsymbol{s}, Q''\boldsymbol{a}) \\ &= \frac{1}{N!}\sum_{Q''\in\mathcal{G}^N} q_{\pi^{Q''}}(\boldsymbol{s}, \boldsymbol{a}) \\ &\propto \bar{\pi}(\boldsymbol{s}, \boldsymbol{a}),\end{aligned}$$

□

B.4 Proof of Theorem 3.4

We start with a simple lemma.

Lemma B.4. *For any $\boldsymbol{w}$ and any $\boldsymbol{v} \in \mathbb{R}^N$, we have that $GGF_{\mathbf{1}/N}[\boldsymbol{v}] \geq \mathrm{GGF}_{\boldsymbol{w}}[\boldsymbol{v}]$.*

Proof. This simply follows from:

$$\mathrm{GGF}_{\mathbf{1}/N}[\boldsymbol{v}] = \frac{1}{N}\mathbf{1}^\top \boldsymbol{v} = \left(\frac{1}{N!}\sum_{Q\in\mathcal{G}^N} Q\boldsymbol{w}\right)^\top \boldsymbol{v} \geq \min_{Q\in\mathcal{G}^N}(Q\boldsymbol{w})^\top \boldsymbol{v} = \mathrm{GGF}_{\boldsymbol{w}}[\boldsymbol{v}].$$

□

Theorem 3.4 (Utilitarian Reduction) *For a symmetric WCMDP (definition 3.1), let $\Pi^*_{\mathbf{1}/N,PI}$ be the set of optimal policies for the utilitarian approach (definition 2.1) that is permutation invariant, then $\Pi^*_{\mathbf{1}/N,PI}$ is necessarily non-empty and all $\pi^*_{\mathbf{1}/N,PI} \in \Pi^*_{\mathbf{1}/N,PI}$ satisfies*

$$\mathrm{GGF}_{\boldsymbol{w}}[\boldsymbol{V}_0^{\pi^*_{\mathbf{1}/N,\mathrm{PI}}}] = \max_{\pi} \mathrm{GGF}_{\boldsymbol{w}}\left[\boldsymbol{V}_0^{\pi}\right], \forall \boldsymbol{w} \in \Delta(N).$$

Proof. Let us denote an optimal policy to the special case of the GGF-WCMDP problem (2) with equal weights as $\pi^*_{\mathbf{1}/N}$:

$$\pi^*_{\mathbf{1}/N} \in \arg\max_{\pi} \mathrm{GGF}_{\mathbf{1}/N}\left[\boldsymbol{V}_0^{\pi}\right]. \tag{10}$$

Based on Lemma 3.3, we can construct a permutation invariant policy $\bar{\pi}^*_{\mathbf{1}/N}$ satisfying

$$\bar{V}_0^{\pi^*_{\mathbf{1}/N}}\mathbf{1} = \boldsymbol{V}_0^{\bar{\pi}^*_{\mathbf{1}/N}}, \tag{11}$$

then with equation (11) and the fact that any weight vector $\boldsymbol{w}$ must sum to 1, we have that

$$\mathrm{GGF}_{\mathbf{1}/N}\left[\boldsymbol{V}_0^{\pi^*_{\mathbf{1}/N}}\right] = \bar{V}_0^{\pi^*_{\mathbf{1}/N}} = \mathrm{GGF}_{\boldsymbol{w}}\left[\frac{1}{N}\sum_{n=1}^{N} V_{0,n}^{\pi^*_{\mathbf{1}/N}}\mathbf{1}\right] = \mathrm{GGF}_{\boldsymbol{w}}\left[\boldsymbol{V}_0^{\bar{\pi}^*_{\mathbf{1}/N}}\right], \forall \boldsymbol{w}.$$

Further, given any $\boldsymbol{w}$, let us denote with $\pi^*_{\boldsymbol{w}}$ any optimal policy to the GGF problem with $\boldsymbol{w}$ weights. One can establish that:

$$\mathrm{GGF}_{\boldsymbol{w}}[\boldsymbol{V}_0^{\pi^*_{\boldsymbol{w}}}] \geq \mathrm{GGF}_{\boldsymbol{w}}\left[\boldsymbol{V}_0^{\bar{\pi}^*_{\mathbf{1}/N}}\right] = \mathrm{GGF}_{\mathbf{1}/N}\left[\boldsymbol{V}_0^{\pi^*_{\mathbf{1}/N}}\right] \geq \mathrm{GGF}_{\mathbf{1}/N}\left[\boldsymbol{V}_0^{\pi^*_{\boldsymbol{w}}}\right]. \tag{12}$$

Considering that the largest optimal value for the GGF problem is achieved when weights are equal (see Lemma B.4):

$$\mathrm{GGF}_{\mathbf{1}/N}\left[\boldsymbol{V}_0^{\pi}\right] \geq \mathrm{GGF}_{\boldsymbol{w}}[\boldsymbol{V}_0^{\pi}], \forall \pi,\ \forall \boldsymbol{w} \in \Delta(N).$$

The inequalities in (12) should therefore all reach equality:

$$\mathrm{GGF}_{\boldsymbol{w}}[\boldsymbol{V}_0^{\pi^*_{w}}] = \mathrm{GGF}_{\boldsymbol{w}}\left[\boldsymbol{V}_0^{\bar{\pi}^*_{\mathbf{1}/N}}\right] = \mathrm{GGF}_{\mathbf{1}/N}\left[\boldsymbol{V}_0^{\pi^*_{\mathbf{1}/N}}\right].$$

This implies that the bar optimal policy constructed from any optimal policy to the utilitarian approach remains optimal for any weights in the GGF optimization problem. Furthermore, it implies that there exists at least one permutation invariant policy that is optimal for the utilitarian approach.

Now, let us take any optimal permutation invariant policy $\pi^*_{\mathbf{1}/N,\text{PI}}$ to the utilitarian problem. The arguments above can straightforwardly be reused to get the conclusion that $\bar{\pi}^*_{\mathbf{1}/N,\text{PI}}$ have the same properties as the originally constructed $\bar{\pi}$. Namely, for all $\boldsymbol{w}$,

$$\text{GGF}_{\boldsymbol{w}}[\boldsymbol{V}_0^{\pi^*_w}] = \text{GGF}_{\boldsymbol{w}}\left[\boldsymbol{V}_0^{\bar{\pi}^*_{\mathbf{1}/N,\text{PI}}}\right] = \text{GGF}_{\mathbf{1}/N}\left[\boldsymbol{V}_0^{\pi^*_{\mathbf{1}/N}}\right],\ \forall \boldsymbol{w} \in \Delta(N).$$

Looking more closely at $\bar{\pi}^*_{\mathbf{1}/N,\text{PI}}$, we observe that for any $\boldsymbol{s}$ and $\boldsymbol{a}$:

$$\begin{aligned}
\bar{\pi}^*_{\mathbf{1}/N,\text{PI}}(\boldsymbol{s},\boldsymbol{a}) &\propto \frac{1}{N!}\sum_{Q'\in\mathcal{G}^N} q_{\pi^*_{\mathbf{1}/N,\text{PI}},Q'}(\boldsymbol{s},\boldsymbol{a}) \\
&= \frac{1}{N!}\sum_{Q'\in\mathcal{G}^N} q_{\pi^*_{\mathbf{1}/N,\text{PI}}}(\boldsymbol{s},\boldsymbol{a}) \\
&= q_{\pi^*_{\mathbf{1}/N,\text{PI}}}(\boldsymbol{s},\boldsymbol{a}) \\
&\propto \pi^*_{\mathbf{1}/N,\text{PI}}(\boldsymbol{s},\boldsymbol{a}).
\end{aligned}$$

Hence, we have that $\bar{\pi}^*_{\mathbf{1}/N,\text{PI}} = \pi^*_{\mathbf{1}/N,\text{PI}}$. This thus implies that the permutation invariant $\pi^*_{\mathbf{1}/N,\text{PI}}$ already satisfied these properties, i.e.,

$$\text{GGF}_{\boldsymbol{w}}[\boldsymbol{V}_0^{\pi^*_w}] = \text{GGF}_{\boldsymbol{w}}\left[\boldsymbol{V}_0^{\pi^*_{\mathbf{1}/N,\text{PI}}}\right] = \text{GGF}_{\mathbf{1}/N}\left[\boldsymbol{V}_0^{\pi^*_{\mathbf{1}/N}}\right],\ \forall \boldsymbol{w} \in \Delta(N).$$

□

C Count Aggregation MDP

The exact form of the count aggregation MDP is obtained as follows.

Feasible Action The set of feasible actions in state $\boldsymbol{x}$ is defined as

$$\mathcal{A}^{(N)}_{g_s}(\boldsymbol{x}) := \{\boldsymbol{u} \mid \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d_k(a)u_{s,a} \le b_k, \forall k \in \mathcal{K}\ ;\ \sum_{a\in\mathcal{A}} u_{s,a} = x_s, \forall s \in \mathcal{S}\}.$$

Reward Function The average reward for all users is defined as

$$\bar{r}_\phi(\boldsymbol{x},\boldsymbol{u}) = \frac{1}{N}\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} u_{s,a}\cdot r(s,a).$$

Transition Probability The transition probability $p^{(N)}_\phi(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u})$ is the probability that the number of MDP in each state passes from $\boldsymbol{x}$ to $\boldsymbol{x}'$ given the action counts $\boldsymbol{u}$. We define the pre-image $f^{-1}(\boldsymbol{x}')$ as the set containing all elements $\boldsymbol{s}' \in \mathcal{S}^{(N)}$ that maps to $\boldsymbol{x}'$, then $p^{(N)}_\phi(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u}) = \sum_{\boldsymbol{s}'\in f^{-1}(\boldsymbol{x}')} p^{(N)}(\boldsymbol{s}'|f^{-1}(\boldsymbol{x}), g_s^{-1}(\boldsymbol{u}))$.

Given the equivalence of transitions within the pre-image set, for arbitrary state-action pair $(\boldsymbol{s},\boldsymbol{a}) \in \phi^{-1}(\boldsymbol{x},\boldsymbol{u})$, the probability of transitioning from $\boldsymbol{x}$ to $\boldsymbol{x}'$ under action $\boldsymbol{u}$ is the sum of the probabilities of all the individual transitions in the original space that correspond to this count aggregation transition. By using the transition probability in the product space, we obtain

$$p^{(N)}_\phi(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u}) = \sum_{\boldsymbol{s}'\in f^{-1}(\boldsymbol{x}')} p^{(N)}(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a}) = \sum_{\boldsymbol{s}'\in f^{-1}(\boldsymbol{x}')}\prod_{n=1}^{N} p_n(s'_n|s_n,a_n), \tag{13}$$

for any $(\boldsymbol{s},\boldsymbol{a})$ such that $\boldsymbol{x} = f(\boldsymbol{s})$ and $\boldsymbol{u} = g_s(\boldsymbol{a})$.

Initial distribution By using a state count representation for symmetric weakly coupled MDPs, we know that $\mathbf{1}^\top\boldsymbol{x} = N$, so the cardinality of the set can be obtained through multinomial expansion that $(s_1+s_2+\cdots+s_N)^S = \sum_{x_1+x_2+\cdots+x_S=N}\frac{N!}{x_1!x_2!\cdots x_S!}s_1^{x_1}s_2^{x_2}\cdots s_N^{x_S}$. Intuitively, the term $s_1^{x_1}s_2^{x_2}\cdots s_N^{x_S}$ can represent distributing N identical objects (in this case, sub-MDPs) into S distinct categories (corresponding to different states). Thus, for each state count $\boldsymbol{x}$, the number of distinct ways to distribute N sub-MDPs into S states such that the counts match $\boldsymbol{x}$ is given by the multinomial coefficient $|f^{-1}(\boldsymbol{x})| = \frac{N!}{x_1!x_2!\cdots x_S!}$. Given the initial distribution $\mu^{(N)}$ is permutation invariant, the

probability of starting from state $\boldsymbol{x}$ in the initial distribution is

$$\mu_f^{(N)}(\boldsymbol{x}) = \sum_{\boldsymbol{s}\in f^{-1}(\boldsymbol{x})} \mu^{(N)}(\boldsymbol{s}) = |f^{-1}(\boldsymbol{x})| \cdot \mu^{(N)}(\bar{\boldsymbol{s}}) = \frac{N!}{x_1!x_2!\cdots x_S!} \cdot \mu^{(N)}(\bar{\boldsymbol{s}}), \forall \boldsymbol{x}, \tag{14}$$

for any $\bar{\boldsymbol{s}}$ such that $f(\bar{\boldsymbol{s}}) = \boldsymbol{x}$.

D Exact Approaches based on Linear Programming

D.1 Optimal Solutions to the GGF-WCMDP problem

First, we recall the dual linear programming (LP) methods to solve the MDP with discounted rewards when the transition and reward functions are known. The formulation is based on the Bellman equation for optimal policy, and is derived in section 6.9.1 in detail by Puterman (2005).

The dual linear programming formulation for addressing the multi-objective joint MDP can be naturally extended to the context of vector optimization:

$$\begin{aligned} \text{v-max} \quad & \sum_{\boldsymbol{s}\in\mathcal{S}^{(N)}} \sum_{\boldsymbol{a}\in\mathcal{A}^{(N)}} \boldsymbol{r}(\boldsymbol{s},\boldsymbol{a}) q(\boldsymbol{s},\boldsymbol{a}) \\ \text{s.t.} \quad & \sum_{\boldsymbol{a}\in\mathcal{A}^{(N)}} q(\boldsymbol{s},\boldsymbol{a}) - \gamma \sum_{\boldsymbol{s}'\in\mathcal{S}^{(N)}} \sum_{\boldsymbol{a}\in\mathcal{A}^{(N)}} q(\boldsymbol{s}',\boldsymbol{a}) p^{(N)}(\boldsymbol{s}|\boldsymbol{s}',\boldsymbol{a}) = \mu^{(N)}(\boldsymbol{s}), \forall \boldsymbol{s}\in\mathcal{S}^{(N)} \\ & q(\boldsymbol{s},\boldsymbol{a}) \geq 0 \qquad \forall \boldsymbol{s}\in\mathcal{S}^{(N)}, \forall \boldsymbol{a}\in\mathcal{A}^{(N)} \end{aligned}, \tag{15}$$

where any $\mu^{(N)}(\boldsymbol{s}) > 0$ can be chosen, but we normalize the weights such that $\sum_{\boldsymbol{s}\in\mathcal{S}^{(N)}} \mu^{(N)}(\boldsymbol{s}) = 1$ can be interpreted as the probability of starting in a given state $\boldsymbol{s}$.

We can now formally formulate the fair optimization problem by combining the GGF operator (Section 2.2) and the scalarizing function on the reward vector in (15):

$$\begin{aligned} \max \quad & \mathrm{GGF}_{\boldsymbol{w}}[\boldsymbol{v}] \\ \text{s.t.} \quad & \boldsymbol{v} = \sum_{\boldsymbol{s}\in\mathcal{S}^{(N)}} \sum_{\boldsymbol{a}\in\mathcal{A}^{(N)}} \boldsymbol{r}(\boldsymbol{s},\boldsymbol{a}) q(\boldsymbol{s},\boldsymbol{a}) \\ & \sum_{\boldsymbol{a}\in\mathcal{A}^{(N)}} q(\boldsymbol{s},\boldsymbol{a}) - \gamma \sum_{\boldsymbol{s}'\in\mathcal{S}^{(N)}} \sum_{\boldsymbol{a}\in\mathcal{A}^{(N)}} q(\boldsymbol{s}',\boldsymbol{a}) p^{(N)}(\boldsymbol{s}|\boldsymbol{s}',\boldsymbol{a}) = \mu^{(N)}(\boldsymbol{s}), \forall \boldsymbol{s}\in\mathcal{S}^{(N)} \\ & q(\boldsymbol{s},\boldsymbol{a}) \geq 0 \qquad \forall \boldsymbol{s}\in\mathcal{S}^{(N)}, \forall \boldsymbol{a}\in\mathcal{A}^{(N)} \end{aligned}. \tag{16}$$

By adding a permutation matrix $\mathcal{Q}$ to replace the permutation applied to the index set, $\mathrm{GGF}_{\boldsymbol{w}}[\boldsymbol{v}]$ is equivalently represented as

$$\mathrm{GGF}_{\boldsymbol{w}}[\boldsymbol{v}] = \inf_{\boldsymbol{\mathcal{Q}}:\boldsymbol{\mathcal{Q}}\geq 0, \sum_i \mathcal{Q}_{ij}=1, \forall j, \sum_j \mathcal{Q}_{ij}=1, \forall i} \sum_{ij} w_i \mathcal{Q}_{ij} v_j. \tag{17}$$

This reformulation relies on $w_1 \geq w_2 \geq \cdots \geq w_N$ to confirm that at the infimum we have $\min_{\sigma} \sum_{n=1}^{N} w_n v_{\sigma(n)}$. Indeed, if w_1 is not assigned to the lowest element of $\boldsymbol{v}$, then one can get a lower value by transferring the assignment mass from where it is assigned to that element to improve the solution. This form is obtained through LP duality on (17):

$$\sup_{\boldsymbol{\nu},\boldsymbol{\lambda}:\lambda_i+\nu_j\leq w_i v_j, \forall i,j} \sum_{i=1}^{N} \lambda_i + \sum_{j=1}^{N} \nu_j,$$

which leads to

$$\begin{aligned} \max_{\boldsymbol{\nu},\boldsymbol{\lambda},\boldsymbol{q}} \quad & \sum_{i=1}^{N} \lambda_i + \sum_{j=1}^{N} \nu_j \\ \text{s.t.} \quad & \lambda_i + \nu_j \leq w_i v_j \quad \forall i,j = 1,\ldots,N \end{aligned}.$$

Dual variable vectors are denoted by $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$. Combining the constraints in (16), we can get the complete dual LP model with the GGF objective (GGF-LP) in (3).

D.2 Solving Count Aggregation MDP by the Dual LP Model

Since the exact model for the count aggregation MDP $\mathcal{M}_\phi$ is obtained (Appendix C), a dual linear programming model is formulated following section 6.9.1 of Puterman (2005), but with count aggregation representation to solve (4):

$$\begin{aligned} \max \quad & \sum_{\boldsymbol{x}\in\mathcal{S}_f^{(N)}} \sum_{\boldsymbol{u}\in\mathcal{A}_{g_s}^{(N)}} \bar{r}_\phi(\boldsymbol{x},\boldsymbol{u}) q_\phi(\boldsymbol{x},\boldsymbol{u}) \\ \text{s.t.} \quad & \sum_{\boldsymbol{u}\in\mathcal{A}_{g_s}^{(N)}} q_\phi(\boldsymbol{x},\boldsymbol{u}) - \gamma \sum_{\boldsymbol{x}'\in\mathcal{S}_f^{(N)}} \sum_{\boldsymbol{u}\in\mathcal{A}_{g_s}^{(N)}} q_\phi(\boldsymbol{x}',\boldsymbol{u}) p_\phi^{(N)}(\boldsymbol{x}|\boldsymbol{x}',\boldsymbol{u}) = \mu_f^{(N)}(\boldsymbol{x}),\ \forall \boldsymbol{x}\in\mathcal{S}_f^{(N)} \\ & q_\phi(\boldsymbol{x},\boldsymbol{u}) \geq 0 \qquad\qquad \forall \boldsymbol{x}\in\mathcal{S}_f^{(N)}, \forall \boldsymbol{u}\in\mathcal{A}_{g_s}^{(N)} \end{aligned} \tag{18}$$

By choosing the initial distribution as $\mu_f^{(N)}$, the optimal solution $q_\phi(\boldsymbol{x},\boldsymbol{u}), \forall \boldsymbol{x},\boldsymbol{u}$ is equivalent to the optimal solution to the corresponding weakly coupled MDP under the transformation ϕ.

E Model simulator

In the learning setting, the deep RL agent interacts with a simulated environment through a model simulator (Algorithm 2), which leads to the next states $\boldsymbol{x}'$ and the average reward $\bar{r}_\phi$ across all coupled MDPs.

Algorithm 2 Simulation of Transition Dynamics

Input: count state $\boldsymbol{x}$, count action $\boldsymbol{u}$, transition probability $p(s'|s,a)$ and reward function $r(s,a)$.
Initialize: next state $\boldsymbol{x}' \leftarrow \mathbf{0}$, average reward $\bar{r}_\phi \leftarrow 0$
for $s = 1,\ldots,S$ **do**
  for $a = 1,\ldots,A$ **do**
    while $u_{s,a} > 0$ **do**
      Sample the next state index $s' \in [S]$ according to the probability distribution $p(\cdot|s,a)$
      $x'_{s'} \leftarrow x'_{s'} + 1$
      $\bar{r}_\phi \leftarrow \bar{r}_\phi + \frac{1}{N}\cdot r(s,a)$
      $u_{s,a} \leftarrow u_{s,a} - 1$
    end while
  end for
end for
Return: next state $\boldsymbol{x}'$, average reward $\bar{r}_\phi$

F Experimental Design

F.1 Parameter setting

This section details the construction of the components used to generate the test instances based on Akbarzadeh and Mahajan (2019), including cost function, transition matrix, and reset probability. This experiment uses a synthetic data generator implemented on our own that considers a system with S states and binary actions ($A = 2$). There are two possible actions: operate (passive action 0) or replace (active action 1). After generating the cost matrix of the size $\mathcal{S}^{(N)} \cdot \mathcal{A}^{(N)} \cdot N$, we normalize the costs to the range [0, 1] by dividing each entry by the maximum cost over all state action pairs. This ensures that the discounted return always falls within the range $[0, \frac{1}{1-\gamma}]$.

Cost function The cost function $c(s)$ for $s \in [S]$ can be defined in five ways, each representing a different cost structure associated with each state:

- **Linear**: In this case, the cost function $c(s) = s - 1$ increases linearly with the state, which means higher states (worse conditions) incur higher costs.
- **Quadratic**: The cost function $c(s) = (s-1)^2$ applies a more severe penalty for higher states compared to linear cost function.
- **Exponential**: The cost function $c(s) = e^{s-1}$ imposes exponentially increasing costs.

- **Ratio of the Replacement Cost Constant (RCCC)**: The cost function $1.5(S-1)^2$ is constant and based on a ratio of 1.5 to the maximum quadratic cost.
- **Random**: The cost function $c(s)$ is generated randomly within a specified range of [0,1].

Transition function The transition matrix for the deterioration action is constructed as follows. Once the machine reaches the S-th state, it remains in that state indefinitely until being reset by an active action with a probability of 1. For the s-th state $s \in [S-1]$, the probability of remaining in the same state at the next step is given by a model parameter $p_m \in [0,1]$, and the probability to the $s+1$-th state is denoted by $1-p_m$.

Reset probability When a replacement occurs, there is a probability p_s that the machine successfully resets to the first state, and a corresponding probability $1-p_s$ of failing to be repaired and following the deterioration rule. In our experiments, we only consider a pure reset to the first state with probability 1.

F.2 Chosen Cost Structures

We consider two cost models to reflect real-world maintenance and operation dynamics:

(i) *Exponential-RCCC*: In this scenario, operational costs increase exponentially with age, and exceed replacement costs in the worst state to encourage replacements. This scenario fits the operational dynamics of transportation fleets, such as drone batteries, where operational inefficiencies grow rapidly and can lead to significant damage to the drones.

(ii) *Quadratic-RCCC*: In contrast to *scenario ii)*, operational costs increase quadratically with machine age, while replacement costs remain constant and always higher than operational costs. This setup is typical for high-valued machinery, where the cost of replacement can be significant compared to operational expenses.

F.3 Hyperparameters

In our experimental setup, we chose PPO algorithm to implement the count-proportion based architecture. The hidden layers are fully connected and the Tanh activation function is used. There are two layers, with each layer consisting of 64 units. The learning rates for the actor is set to 5×10^{-4} and the critic is set to 3×10^{-4}.

F.4 Additional Results on LP Solving Times

The results in Tables 3 and 4 provide details on solving the GGF-LP model and the Count dual LP model on the Quadratic-RCCC instances as the number of machines N increases from 2 to 7. The state size is set to $S = 3$, the action size to $A = 2$, and the resource to $b = 1$. The first block of the tables shows the number of constraints and variables. The second block provides the model solving times, with the standard deviations listed in parentheses. The results are based on 5 runs. Notice that the LP solve time inludes pre-solve, the wallclock time, and post-solve times. The wallclock time is listed separately to highlight the difference from the pure LP Solve time, but not included in the total time calculation.

Table 3: Statistics for GGF-LP Model (3).

	$N = 2$	$N = 3$	$N = 4$	$N = 5$	$N = 6$	$N = 7$
# Constraints	13	36	97	268	765	2236
# Variables	31	114	413	1468	5115	17510
Data Build (s)	0.0019	0.0085	0.1076	1.3698	17.7180	320.0141
	(0.00)	(0.00)	(0.01)	(0.05)	(0.52)	(16.85)
LP Build (s)	0.0028	0.0185	0.1449	1.4673	20.7464	392.0150
	(0.00)	(0.01)	(0.01)	(0.08)	(4.00)	(33.97)
LP Solve (s)	0.0187	0.0212	0.1846	1.2914	13.0377	138.1272
	(0.02)	(0.00)	(0.06)	(0.09)	(0.63)	(2.33)
Wallclock Solve* (s)	0.0026	0.0018	0.0115	0.0493	0.7849	13.3167
	(0.00)	(0.00)	(0.00)	(0.00)	(0.16)	(0.19)
LP Extract (s)	0.0022	0.0019	0.0044	0.0158	0.0711	0.2828
	(0.00)	(0.00)	(0.00)	(0.00)	(0.01)	(0.03)
Total Time (s)	0.0256	0.0500	0.4416	4.1443	51.5732	864.4391
	(0.03)	(0.01)	(0.06)	(0.07)	(3.87)	(56.75)

*Wall clock solve time is included in the LP Solve time.

Table 4: Statistics for Count dual LP Model (D.2).

	$N = 2$	$N = 3$	$N = 4$	$N = 5$	$N = 6$	$N = 7$
# Constraints	6	10	15	21	28	36
# Variables	24	40	60	84	112	144
Data Build (s)	0.0035	0.0134	0.1306	1.4401	17.4080	204.4550
	(0.00)	(0.00)	(0.01)	(0.01)	(0.40)	(4.49)
LP Build (s)	0.0018	0.0031	0.0056	0.0081	0.0297	0.0501
	(0.00)	(0.00)	(0.01)	(0.00)	(0.01)	(0.01)
LP Solve (s)	0.0375	0.0362	0.0466	0.0386	0.0745	0.1634
	(0.02)	(0.02)	(0.00)	(0.01)	(0.01)	(0.06)
Wallclock Solve* (s)	0.0034	0.0053	0.0031	0.0026	0.0034	0.0105
	(0.00)	(0.00)	(0.00)	(0.00)	(0.00)	(0.00)
LP Extract (s)	0.0053	0.0025	0.0687	0.0026	0.0051	0.0209
	(0.00)	(0.00)	(0.00)	(0.00)	(0.00)	(0.00)
Total Time (s)	0.0481	0.0552	0.2515	1.4894	17.5173	204.6893
	(0.02)	(0.02)	(0.13)	(0.01)	(0.42)	(4.44)

*Wall clock solve time is included in the LP Solve time.