Strategic bidding in freight transport using deep reinforcement learning

W. J. A. van Heeswijk¹

Accepted: 21 January 2022
© The Author(s) 2022

Abstract

This paper presents a multi-agent reinforcement learning algorithm to represent strategic bidding behavior by carriers and shippers in freight transport markets. We investigate whether feasible market equilibria arise without central control or communication between agents. Observed behavior in such environments serves as a stepping stone towards self-organizing logistics systems like the Physical Internet, while also offering valuable insights for the design of contemporary transport brokerage platforms. We model an agent-based environment in which shipper and carrier actively learn bidding strategies using policy gradient methods, posing bid- and ask prices at the individual container level. Both agents aim to learn the best response given the expected behavior of the opposing agent. Inspired by financial markets, a neutral broker allocates jobs based on bid-ask spreads. Our game-theoretical analysis and numerical experiments focus on behavioral insights. To evaluate system performance, we measure adherence to Nash equilibria, fairness of reward division and utilization of transport capacity. We observe good performance both in predictable, deterministic settings ($\sim 95\%$ adherence to Nash equilibria) and highly stochastic environments ($\sim 85\%$ adherence). Risk-seeking behavior may increase an agent's reward share, yet overly aggressive strategies destabilize the system. The results suggest a potential for full automation and decentralization of freight transport markets. These insights ease the design of real-world market platforms, suggesting an innate tendency of markets to reach equilibria without behavioral models, information sharing or explicit incentives.

Keywords Strategic bidding · Deep reinforcement learning · Policy gradient · Self-organizing logistics

1 Introduction

While the transport sector traditionally relies on fixed contracts and manual negotiations, modern technology allows for automated negotiation and more appropriate responses to its inherently dynamic nature. Paradigms such as the Physical Internet and self-organizing logistics provide conceptual outlines to organize such systems, yet many facets remain unexplored.

✉ W. J. A. van Heeswijk
w.j.a.vanheeswijk@utwente.nl

¹ University of Twente, Drienerlolaan 5, Enschede, The Netherlands

This paper investigates a strategic bidding mechanism based on multi-agent reinforcement learning, deliberately exploring a setting without communication or centralized control and presenting decentralized planning in a pure form. For present-day applications, the resulting insights are useful for market platform design, which deal with challenges such as horizontal collaboration, information-sharing infrastructures and incentive design (Cruijssen 2020; Karam et al. 2021). With the platform economy rapidly growing, finding ways to connect independent market participants becomes increasingly important (Kenney and Zysman 2016).

We study a multi-agent environment containing a carrier, shipper and broker, as envisioned in the Physical Internet (Montreuil et al. 2013). For each individual transport job (e.g., a smart container), the shipper and carrier pose bid- and ask prices respectively. The neutral broker agent—with a business model inspired by transport matching platforms and highly decentralized financial markets—matches these bids and asks at a batch level, solving a knapsack problem that maximizes bid-ask spread. Being a rule-based agent, the broker may be considered as the ‘environment’ rather than an active agent. The shipper and carrier compete against each other, actively learning strategies to maximize their own reward given the deployed strategy of the opponent. We stress that the learning algorithm is model-free; agents do not explicitly model the expected strategy of the opponent. By investigating a model-free and stochastic setting, we minimize behavioral assumptions. Both agents constantly adjust their strategy in an online learning setting, representing a self-organizing freight transport market. The behavior of such a market is the key interest of this work, keeping future extensions towards multiple carriers and shippers in mind. Inspired by game theory, efficiency and fairness are the primary performance metrics.

This paper expands upon Van Heeswijk (2019)—which introduced a learning bidding agent—in several ways. First, the carrier, which was a passive price-taker in the earlier work, now is a learning agent with a dynamic strategy. This adds a strategic dimension that vastly increases the complexity. Second, we add a (non-learning) broker agent to the setting, exploring the role of a neutral transport planner in self-organizing systems. Third, we test a variety of actor-critic models and deep learning techniques, whereas the earlier work relies on a basic policy gradient algorithm. Fourth, we root the performance metrics and numerical results in game-theoretical foundations.

The remainder of this paper is structured as follows. Section 2 discusses related literature. In Sect. 3 we define the system as a Markov Decision Process (MDP) model. Section 4 presents the solution method; we describe several variants of policy gradient algorithms to learn bidding- and asking prices. Section 5 presents the experimental design and Sect. 6 the numerical results. Section 7 ends the paper with the main conclusions.

2 Literature review

This literature overview is structured as follows. First, we discuss the core concepts of self-organizing logistics and the Physical Internet. Second, we highlight several studies on reinforcement learning in the Delivery Dispatching Problem, which relates to our setting. Third, we discuss optimal bidding and (contemporary) transport platforms. Fourth, we assess the links with game theory, which will be used for our analysis and experimental design.

The inspiration for this paper stems from the Physical Internet paradigm (Montreuil 2011; Montreuil et al. 2013). The Physical Internet envisions an open market in which logistics services are offered, with automated interactions between smart containers and other constituents determining routes and schedules. Sallez et al. (2016) emphasize the active role of

smart containers, communicating, memorizing, negotiating, and learning both individually and collectively. Ambra et al. (2019) present a recent literature review of work performed in the domain of the Physical Internet. Interestingly, their overview does not mention works that define the smart container itself as an actor. Instead, existing works focus on traditional actors such as carriers, shippers and logistics service providers, even though smart containers supposedly route themselves in the Physical Internet.

The problem studied in this paper is related to the Delivery Dispatching Problem (Minkoff 1993), which entails dispatching decisions from a carrier's perspective. Transport jobs arrive at a hub according to some external stochastic process. The carrier subsequently decides which subset of jobs to accept, anticipating future job arrivals. Basic instances may be solved with queuing models, but more complicated variants are computationally intractable; researchers often resort to reinforcement learning to learn high-quality strategies. We highlight some recent works in this domain. Klapp et al. (2018) develop an algorithm that solves the dispatching problem for a transport service operating on the real line. Van Heeswijk and La Poutré (2018) compare centralized and decentralized transport for networks with fixed line transport services, concluding that decentralized planning yields considerable computational benefits. Van Heeswijk et al. (2019a) study a variant including a routing component, using value function approximation to find strategies. Voccia et al. (2019) solve a variant that includes both pickups and deliveries. Our current paper distinguishes itself from the aforementioned works by assigning jobs based on bid-ask spreads (neutral perspective) rather than transport efficiency (carrier perspective).

Next, we highlight related works on optimal bidding in freight transport; in most of these studies competing carriers bid on transport jobs. For instance, Yan et al. (2018) propose a particle swarm optimization algorithm to place carrier bids on jobs. Miller and Nie (2020) emphasize the importance of integrating carrier competition, routing and bidding. Wang et al. (2018) design a reinforcement learning algorithm based on knowledge gradients, solving for a bidding structure with a broker intermediating between carriers and shippers. The broker aims to propose a price that satisfies both carrier and shipper, taking a percentage of accepted bids as its reward. In the context of market platforms, Zha et al. (2017) study market equilibriums, concluding that carriers and the broker benefit in times of scarce supply. Atasoy et al. (2020) address interaction between a brokerage platform and multiple carriers, finding that the broker retains profitability even after providing financial incentives to carriers. In a Physical Internet context, Qiao et al. (2019) model hubs as spot freight markets where carriers can place bids on transport bids. To this end, they propose a dynamic pricing model based on an auction mechanism. Most studies assume that shippers have limited to no influence in the bidding process; we aim to add a fresh perspective with this work. This paper builds onto the work of Van Heeswijk (2019), in which the shipper is a learning agent and the carrier a passive price taker. The author uses a policy gradient algorithm to learn bidding strategies. To the best of our knowledge, there are no studies that explicitly model both carriers and shippers as intelligent bidding agents.

We analyze our experimental results from a game theoretical perspective. Conceptually, the bid-ask problem may be classified as an infinitely repeated non-cooperative game, in which both agents aim to maximize their average reward. More specifically, it may be classified as a bargaining game such as defined by Nash (1953), in which both agents claim a share of system-wide gains. Folk theorems provide insights into equilibria in such settings (Friedman 1971). Each payoff profile that is feasible and individually rational in the one-shot game constitutes a Nash equilibrium in the repeated game. For bargaining games, the presence of a threat or disagreement point for deviating opponents is essential to prove the existence of Nash equilibria. Aumann and Shapley (1994) and Rubinstein (1980, 1994) present solutions for

Nash equilibria under temporary punishments. Performance metrics and normative solutions are discussed in Sect. 5.2.

3 System description

This section formally defines our system as a Markov Decision Process model. In Sect. 3.1 we provide a high-level outline of the system and the agents involved. Section 3.2 describes the system state; Sect. 3.3 follows up with the decisions and reward functions. Finally, Sect. 3.4 provides the transition function to complete the model definition.

3.1 Model outline

The system contains three agent types: (i) shipper (S), (ii) carrier (C) and (iii) broker (B). We provide a global outline here; more detail follows in the subsequent sections.

We consider a singular transport service with a fixed origin (e.g., a transport hub). Every day, the *shipper* places individual bids (autonomous smart containers using a joint stochastic policy) for each job to be transported to its destination on a real line. The job is shipped if its bid is accepted. If the job reaches its due date and is still not shipped, it is removed from the system as a failed job. Second, the *carrier* is responsible for performing the transport service. Without knowing the bid price, it poses a daily ask price for each job. Depending on volume and distance, each job has its own marginal transport costs. When the broker assigns a job, the carrier is obliged to transport it and receives the requested ask price. Third, the freight *broker* is responsible for scheduling jobs. After receiving all bids and asks for the day, the broker assigns transport jobs to the carrier. Its profit is the difference between the bid- and ask price of each shipped job. This means that (i) jobs are never shipped when the ask exceeds the bid and (ii) in case transport capacity is insufficient, the broker assigns jobs in a way that maximizes its own total profit. Unlike the other agents, the broker's behavior is rule-based rather than learned (i.e., a passive agent). Furthermore, it makes decisions at the batch level (scheduling all jobs for the day) rather than at the individual job level. We illustrate the strategic bidding problem in Fig. 1.

We consider an infinite horizon setting in which strategies are updated daily (i.e., an online learning setting). For practical and notational purposes, the horizon length is set to $N = \infty$ and the corresponding decision epochs (representing days) to $n \in \{0, 1, \dots, N\}$. Every day n , a transport service with fixed capacity ζ^C departs along the real line; all shipped jobs are delivered the same day (i.e., before the next epoch). Neither past bid- and ask prices nor job allocations impact current decisions, satisfying the Markovian memoryless property.

3.2 State description

The system state is defined as the set of transport jobs. An individual job is defined by an attribute vector j . In addition to the global time horizon that runs till N , each job has an individual time horizon $\mathcal{T}_{n,j}$ that corresponds to the number of decision epochs till due date (i.e., decreasing with time). Each job is represented by an attribute vector:

$$j = \begin{pmatrix} j_\tau = \# \text{ epochs till due date} \\ j_d = \text{distance to destination} \\ j_v = \text{container volume} \end{pmatrix}$$

Fig. 1 Visual representation of the bid-ask system. For each job, the shipper and carrier pose a bid- and ask price respectively. The broker assigns jobs based on bid-ask spread. For the carrier and shipper, the process is essentially a black box

The integer attribute $j_\tau \in [0, \tau^{max}]$ indicates how many decision epochs remain until the latest possible shipment date. Whenever a new job arrives in the system, we initialize $\mathcal{T}_{n,j} = \{j_\tau, j_\tau - 1, \dots, 0\}$. The time horizon is tied to the individual job; the corresponding number of decision epochs till due date—represented by attribute j_τ —is decremented with each time step. If $j_\tau = 0$ and the job has not been shipped, it is removed from the system. Next, the attribute $j_d \in [1, d^{max}]$ indicates the distance between origin and destination. Finally, the job volume $j_v \in [1, \zeta^{max}]$, with $\zeta^{max} \leq \zeta^C$ represents the transport capacity required for the job. The system state at day n is represented by the set $\mathbf{J}_n$, which contains all jobs $\mathbf{j}$ present in the system. We use $\mathcal{J}_n$ to denote the set of feasible states.

3.3 Decisions and rewards

We introduce the decisions and reward functions. For each job, the shipper places a bid $b_{n,j} \in \mathbb{R}$ and the carrier poses an ask price $a_{n,j} \in \mathbb{R}$. The broker assigns jobs to the carrier, earning $b_{n,j} - a_{n,j}$ for each shipped job. As strategic behavior is the focal point of interest in this study, for each job only one bid/ask per epoch may be posed.

We first discuss the decisions and reward function of the shipper. The shipper bids according to a strategy $\pi_n^{ship} = \mathbb{P}(b_{n,j} \mid (\mathbf{j}, \mathbf{J}_n))$ —used by all smart containers—with Π^S denoting the set of feasible strategies. How to obtain a strategy will be explained in Sect. 4. All bids are stored in a vector $\mathbf{b}_n = [b_{n,j}]_{\forall j \in \mathbf{J}_n}$. Remind that the payoff depends on the broker's decision—denoted by binary variable $\gamma_{n,j}$, with 1 indicating the job will be shipped—and is unknown when posing the bid. The maximum willingness to pay for transporting job j is represented by $c_j^{S,max} = c^{S,max} \cdot j_v \cdot j_d$ (i.e., depending on volume and distance); this value is used to compute the reward. As a well-functioning system is desired, we add variable penalties to express regret on failed bids, with lower bids yielding higher penalties (note that reward functions of shippers and carriers are not symmetric). At any given decision epoch, the direct reward function for individual jobs is defined as follows:

$$r_j^S(\gamma_{n,j}, b_{n,j}) = \begin{cases} c_j^{S,max} - b_{n,j} & \text{if } \gamma_{n,j} = 1 \\ -\max(0, c_j^{S,max} - b_{n,j}) & \text{if } \gamma_{n,j} = 0 \end{cases} . \quad (1)$$

The carrier makes its decision according to a strategy $\pi_n^C = \mathbb{P}(a_{n,j} \mid (j, J_n))$, with Π^C denoting the set of strategies. Ask prices during an episode are stored in a vector $\mathbf{a}_n$. Each job has a marginal transport cost that depends on its distance and volume: $c_j^{C, trn} = c_j^{C, trn} \cdot j_v \cdot j_d$. Posing an ask price below the transport costs yields a loss when accepted. A penalty is incurred when (i) the ask price exceeds the shipment costs and (ii) there is sufficient idle capacity to accommodate the rejected job. If a job is assigned, the carrier's reward is the difference between the ask price and the transport costs:

$$r_j^C(\gamma_{n,j}, a_{n,j}) = \begin{cases} a_{n,j} - c_j^{C, trn} & \text{if } \gamma_{n,j} = 1 \\ -\max(0, a_{n,j} - c_j^{C, trn}) & \text{if } \gamma_{n,j} = 0 \wedge \zeta^C - \left(\sum_{j \in J_n} \gamma_{n,j} \cdot j_v\right) > 0 \\ 0 & \text{otherwise} \end{cases} . \quad (2)$$

For the broker, a job's value is its bid price ($b_{n,j}$) minus its ask price ($a_{n,j}$). Using these values and the available capacity, the broker solves a 0–1 knapsack problem for the entire batch of jobs, using dynamic programming (Kellerer et al. 2004). The broker maximizes its direct rewards by selecting γ_n as follows:

$$\arg \max_{\gamma_n \in \Gamma(J_n)} \left(\sum_{j \in J_n} \gamma_{n,j} (b_{n,j} - a_{n,j}) \right) , \quad (3)$$

s.t.

$$\sum_{j \in J_n} \gamma_{n,j} \cdot j_v \leq \zeta^C .$$

The corresponding reward function for the broker is:

$$r_j^B(\gamma_{n,j}, b_{n,j}, a_{n,j}) = \begin{cases} b_{n,j} - a_{n,j} & \text{if } \gamma_{n,j} = 1 \\ 0 & \text{otherwise} \end{cases} . \quad (4)$$

Note that the broker never selects jobs with a higher ask than bid; this would yield a negative payoff. We formalize this minor result in Lemma 1, serving as a building block for later proofs.

Lemma 1 (Job selection when bid price is lower than ask price) *If $b_{n,j} < a_{n,j}$, the broker will always set $\gamma_{n,j} = 0$ to maximize its profits.*

Proof The proof is found in “Appendix A”. The broker always rejects jobs that would yield a negative payoff, since rejecting yields a payoff of 0. $\square$

3.4 Transition function

To conclude the system definition, we describe the transition function $X: (J_n, \omega_{n+1}, \gamma_n) \mapsto J_{n+1}$ —formally outlined in Algorithm 1—for the system state that occurs in the time step from decision epoch n to $n + 1$. Three state changes occur during a time step; (i) number of decision epochs till due dates are decreased for all unshipped jobs, (ii) failed and shipped jobs are removed and (iii) newly arrived jobs (denoted by set $\omega_{n+1} \in \Omega$) are added.

Algorithm 1 Transition function $X(J_n, \omega_{n+1}, \gamma_n)$

0:	Input: $J_n, \omega_{n+1}, \gamma_n$	► Current state, job arrivals, shipping selection
1:	$J_{n+1} \leftarrow \emptyset$	► Initialize next state
2:	$J_n^x \leftarrow J_n$	► Copy state
3:	$\forall j \in J_n$	► Loop over all jobs
4:	$J_n^x \leftarrow J_n^x \setminus j \mid \gamma_{n,j} = 1$	► Remove shipped job
5:	$J_n^x \leftarrow J_n^x \setminus j \mid j_\tau = 0 \wedge \gamma_{n,j} = 0$	► Remove unshipped job at due date
6:	$j_\tau \leftarrow j_\tau - 1 \mid j_\tau > 0$	► Decrement number of epochs till due date
7:	$J_{n+1} \leftarrow J_n^x \cup \omega_{n+1}$	► Merge existing and new job sets
8:	Output: J_{n+1}	► New state

3.5 Policies and game-theoretical properties

From a game-theoretical perspective, the system has some interesting properties that are worth exploring before turning to solutions. In infinitely repeated games without discounting (corresponding to our infinite horizon problem), a common objective is to maximize average profits (Friedman 1971). Following the limit of means theorem, the optimal strategy (for the shipper, carrier is near-equivalent) looks as follows:

$$\pi^{S,*} = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{n=0}^N \sum_{j \in J_n} \left(\sum_{n'=n}^{n+|T_j|} \arg \max_{b_{n',j} \in \mathbb{R}} \mathbb{E}(r_j(\gamma_{n',j}, b_{n',j})) \right) \quad (5)$$

The problem may be classified as a non-cooperative bargaining game (Nash 1953), in which the difference between transport costs and maximum willingness to pay is a surplus to be divided between shipper and carrier (remind the broker does not actively influence the game). Both agents independently ‘claim’ a proportion of the system-wide gain. The so-called feasibility set contains all solutions in which both agents achieve a nonnegative payoff. If no agreement is reached (i.e., the cumulative claim exceeds the system-wide gain or agents only earn negative payoffs), a disagreement point should exist. That is, each agent can execute a credible threat when the opponent deviates from the set of feasible solutions, as such capping its payoff. As shown in Lemma 2, each agent may cap the opponent’s payoff at 0. We prove this result for the carrier; a similar proof can be constructed for the shipper.

Lemma 2 (Existence of disagreement point) *For the carrier, there exists a strategy π^C that ensures the shipper’s payoff r_j^S equals at most 0 for any job j , regardless of the opponent’s strategy π^S .*

Proof The full proof is found in “Appendix A”. If the carrier asks a price higher than the shipper’s maximum willingness to pay, the shipper must either bid higher than that (resulting in a negative payoff when accepted) or bid below the ask and forfeit the agreement (payoff of 0). $\square$

Following Nash (1953), uncertainty vanishes in the limit and the game converges to an equilibrium. In such an equilibrium, agents cannot improve their payoff by unilaterally changing their strategy. The folk theorem (Friedman 1971) states that an equilibrium payoff profile

should satisfy two properties. First, the payoff should be a convex combination of payoffs of the stage game, e.g., a weighted average as defined in Eq. (5). Second, the equilibrium payoff must be individually rational, paying at least as much as the deviation point. Lemma 3 shows that the latter condition holds and formalizes the Nash equilibrium.

Lemma 3 (Existence of Nash equilibrium) *Any payoff profile satisfying $r_j^C + r_j^S \equiv c_j^{S,max} - c_j^{C, trn}$ is a Nash equilibrium. This payoff profile is achieved by any pair of strategies satisfying $c_j^{C, trn} \leq a_{n,j} = b_{n,j} \leq c_j^{S,max}$.*

Proof The full proof is found in “Appendix A”. Intuitively, when $a_{n,j} < b_{n,j}$, an agent could unilaterally improve its payoff without triggering the disagreement point. All profiles not satisfying $c_j^{C, trn} \leq a_{n,j} = b_{n,j} \leq c_j^{S,max}$ are outside the feasible set and invoke the disagreement point. $\square$

The key result of this section is that any payoff profile satisfying $c_j^{C, trn} \leq a_{n,j} = b_{n,j} \leq c_j^{S,max}$ is a Nash equilibrium. Although the bargaining game is non-cooperative, several *normative* solutions exist that incorporate utility symmetry and Pareto efficiency as fairness criteria. The Nash bargaining *solution* (not be confounded with the equilibrium) describes the product of utilities as optimal (Nash 1950), the Kalai–Smorodinsky solution maximizes utility ratios (Kalai and Smorodinsky 1975), and Kalai’s solution maximizes the minimum surplus utility (Kalai 1977). In Sect. 5 we use the game-theoretical foundation to define appropriate measures for the experimental design.

4 Solution method

This section explains the solution method, which is based on deep reinforcement learning techniques. Finding optimal strategies to solve Eq. (5) is hard. The expectation depends on both the (unknown and potentially mixed) strategy of the opposing agent and the stochastic realization of new jobs; the optimal strategy today might fail tomorrow. Furthermore, bid and ask prices are generated at the individual job level, whereas the broker allocates jobs at the batch level. Finally, continuous action spaces by definition contain infinitely many actions. Thus, we use reinforcement learning to learn approximate strategies. Both shipper and carrier actively learn their pricing strategies based on observations. As changing the strategy of one agent influences the other, we are dealing with a highly non-stationary system.

In Sect. 4.1 we present a stochastic policy gradient algorithm, in which the strategy is constantly adjusted to maximize expected rewards given the expected value. Section 4.2 presents several extensions to the base algorithm, including value function approximations (actor-critic models). Finally, Sect. 4.3 describes the update procedure.

4.1 Policy gradient learning

The reinforcement learning algorithms used in this paper are policy gradient algorithms, operating directly on the strategy. The strategy itself is stochastic, meaning the bid/ask is drawn from a distribution. This makes the strategy harder to counter than deterministic or rule-based strategies. In addition, stochastic policies tend to work better in uncertain environments; the opponent’s strategy is never known with certainty. Policy gradient methods can be expanded by adding a value function approximation to estimate downstream rewards

related to current actions—in that case we speak of actor-critic algorithms. We first discuss the vanilla policy gradient algorithm REINFORCE (Williams 1992) and extend to actor-critic models in Sect. 4.2. For readability, we only present notation for the shipper; carrier variants are near-identical.

Learning takes place in episodes $m \in \{0, 1, \dots, M\}$, with each episode containing a finite number of decision epochs n . Every episode m yields a batch of job observations with corresponding rewards for shipper and carrier, used to update the policies. Rewards are weighted equally and thus not discounted.

In policy gradient reinforcement learning, actions are applied directly on the state. The actions are determined by a stochastic strategy:

$$\pi_{\theta^S}^{S,m}(b_{n,j}^m \mid \mathbf{j}, \mathbf{J}_n) = \mathbb{P}^{\theta^S}(b_{n,j}^m \mid \mathbf{j}, \mathbf{J}_n), \quad (6)$$

where θ^S is the parametrization of the strategy. The probability distributions representing the strategies are Gaussian distributions; bids (asks) are drawn independently for each container:

$$b_{n,j}^m \sim \mathcal{N}(\mu_{\theta^S}(\mathbf{j}, \mathbf{J}_n), \sigma_{\theta^S}), \forall \mathbf{j} \in \mathbf{J}_n. \quad (7)$$

The parameterized standard deviation σ_{θ^S} determines the level of exploration while learning, but is also an integral part of the strategy itself. Standard deviations typically decrease to small values once appropriate means are identified, but may also be larger when retaining some fluctuation is beneficial. For instance, strategies embedding randomness are more difficult to counter.

The stored action-reward trajectories during each episode m indicate which actions resulted in good rewards. We compute gradients which ensure strategy updates in that direction. Only completed jobs (i.e., shipped or removed) are used to update the strategy, such that we capture full reward trajectories. We might also learn from single observations (i.e., uncompleted jobs), yet full reward trajectories are unbiased and the trajectories are fairly short in this problem setting. Completed jobs are stored in a set $\hat{\mathbf{J}}^m$. For each episode the cumulative rewards per job—shown for the shipper here, with a slight abuse of notation—are defined as follows:

$$\hat{v}_{n,j}^{S,m}(\gamma_{n,j}, b_{n,j}) = \begin{cases} r_{n,j}(\gamma_{n,j}, b_{n,j})^{S,m} & \text{if } j_\tau = 0 \\ r_{n,j}(\gamma_{n,j}, b_{n,j})^{S,m} + \hat{v}_{n+1,j}^{S,m} & \text{if } j_\tau > 0 \end{cases}, \forall j_\tau \in \mathcal{T}_j.$$

The cumulative rewards observed at time n in episode m are stored in vectors $\hat{\mathbf{v}}_n^{S,m} = [\hat{v}_{n,j}^{S,m}]_{j_\tau \in \mathcal{T}_j}]_{\mathbf{j} \in \mathbf{J}_n}$ and $\hat{\mathbf{v}}_n^{C,m} = [\hat{v}_{n,j}^{C,m}]_{j_\tau \in \mathcal{T}_j}]_{\mathbf{j} \in \mathbf{J}_n}$. At the end of each episode, we can construct the information vector:

$$\mathbf{I}^{S,m} = \left[[\mathbf{J}_n^m, \mathbf{b}_n^m, \hat{\mathbf{v}}_n^{S,m}, \boldsymbol{\gamma}_n^m]_{\forall n \in \{0, \dots, N\}}, \hat{\mathbf{J}}^m \right].$$

The information vector contains the states, actions and rewards required for the strategy updates. For this purpose we utilize the policy gradient theorem; see Sutton and Barto (2018) for a detailed description. We present the theorem for the shipper:

$$\nabla_{\theta^S} \pi_{\theta^S}^{S,m} \propto \sum_{n=1}^N \left(\int_{\mathbf{J}_n \in \mathcal{J}_n} \mathbb{P}^{\pi_{\theta^S}}(\mathbf{J}_n \mid \mathbf{J}_{n-1}) \int_{b_{n,j}^m \in \mathbb{R}} \nabla_{\theta^S} \pi_{\theta^S}^S(b_{n,j}^m \mid \mathbf{j}, \mathbf{J}_n) v_{n,j}^{\pi_{\theta^S}^S}(\gamma_{n,j}, b_{n,j}^m) \right).$$

We proceed to apply the policy gradient theorem to our system, adopting a Gaussian decision-making strategy, using a neural network (actor network) to output the distribution parameters.

The carrier and shipper both deploy a neural network, which is utilized at the individual container level (i.e., a jointly used policy). Let θ^S define the set of weight parameters describing the decision-making strategy $\pi_{\theta^S}: (j, J_n) \mapsto b_{n,j}^m$. Furthermore, let $\phi(j, J_n)$ be a feature vector distilling the most salient state attributes, for instance the average number of epochs till due date or the number of jobs waiting (note that information is shared between containers). The features used for our study are described in Sect. 5.4. For the actor network, the feature vector ϕ is the input, θ^S represents the network weights, and the mean bid (ask) μ_{θ^S} and standard deviation σ_{θ^S} are the output. We formalize the strategy as

$$\pi_{\theta^S} = \frac{1}{\sqrt{2\pi}\sigma_{\theta^S}} e^{-\frac{(b_{n,j}^m - \mu_{\theta^S}(j, J_n))^2}{2\sigma_{\theta^S}^2}}, \text{ with } b_{n,j}^m \text{ being the bid price, } \mu_{\theta^S}(j, J_n) \text{ the Gaussian mean and } \sigma_{\theta^S} \text{ the parametrized standard deviation.}^1$$

The corresponding action $b_{n,j}^m$ is acquired from the inverse normal distribution. Parameter updates take place after each episode, utilizing a function $U(\theta^{S,m}, I^{S,m})$ that is detailed in Sect. 4.3.

The core concept behind the policy gradient algorithm is that the strategy converges to a price distribution appropriate for the state. Actions with high rewards and low probabilities yield the strongest update signals. The algorithmic procedure to update the parametrized strategy is formalized in Algorithm 2. To summarize the procedure: we perform M training episodes containing N decision epochs each, with new jobs arriving stochastically each epoch. For every individual job, bids and asks are generated at each epoch by the actor networks. Based on bid-ask pairs, the broker allocates jobs by solving a knapsack problem. Reward trajectories per job are stored; if a job is shipped or reaches its due date, it is added to the set of completed jobs. After each episode m , the reward trajectories of all completed job are used to update the actor networks for shipper and carrier respectively.

Algorithm 2 Outline of the policy gradient algorithm (based on Williams 1992)

0:	Input: $\pi_{\theta^S}^0, \pi_{\theta^C}^0$	► Differentiable parametrized strategies
1:	$\theta^S \leftarrow \mathbb{R}^{ \theta^S }, \theta^C \leftarrow \mathbb{R}^{ \theta^C }$	► Initialize parameters
2:	$\forall m \in \{0, \dots, M\}$	► Loop over episodes
3:	$\hat{J}^m \leftarrow \emptyset$	► Initialize completed job set
4:	$J_0 \leftarrow \mathcal{J}_0$	► Generate initial state
5:	$\forall n \in \{0, \dots, N\}$	► Loop over finite time horizon
6:	$b_{n,j}^m \leftarrow \pi_{\theta^S}^m(j, J_n), \forall j \in J_n$	► Bid placement jobs
7:	$a_{n,j}^m \leftarrow \pi_{\theta^C}^m(j, J_n), \forall j \in J_n$	► Ask placement jobs
8:	$\gamma_n^m \leftarrow \arg \max_{\gamma_n^m \in \Gamma(J_n)} \left(\sum_{j \in J_n} \gamma_{n,j}^m (b_{n,j}^m - a_{n,j}^m) \right)$	► Job allocation broker, Eq. (3)
9a:	$\hat{v}_{j,n,j}^{S,m} \leftarrow r_j(\gamma_{n,j}^m, b_{n,j}^m), \forall j \in J_n$	► Compute cumulative rewards shipper
9b:	$\hat{v}_{j,n,j}^{C,m} \leftarrow r_j(\gamma_{n,j}^m, b_{n,j}^m), \forall j \in J_n$	► Compute cumulative rewards carrier
10:	$\forall j \in J_n \mid j_{\tau} = 0 \vee \gamma_{n,j}^m = 1$	► Loop over completed jobs
11:	$\hat{J}^m \leftarrow \hat{J}^m \cup \{j\}$	► Update set of completed jobs
12:	$\omega_{n+1} \leftarrow \Omega$	► Generate job arrivals

¹ π_{θ^S} is the parameterized strategy, π represents the mathematical constant.

13:	$J_{n+1} \leftarrow S^M(J_n, \omega_{n+1}, \gamma_n^m)$	► Transition function, Algorithm 1
14a:	$I^{S,m} \leftarrow [J_n^m, b_n^m, \hat{v}_n^{S,m}, \gamma_n^m]_{\forall n \in \{0, \dots, N\}}, \hat{J}^m$	► Store information shipper
14b:	$I^{C,m} \leftarrow [J_n^m, a_n^m, \hat{v}_n^{C,m}, \gamma_n^m]_{\forall n \in \{0, \dots, N\}}, \hat{J}^m$	► Store information carrier
15a:	$\theta^{S,m+1} \leftarrow U(\theta^{S,m}, I^{S,m})$	► Update actor network shipper
15b:	$\theta^{C,m+1} \leftarrow U(\theta^{C,m}, I^{C,m})$	► Update actor network carrier
16:	Output: $\pi_{\theta^S}^M, \pi_{\theta^C}^M$	► Return tuned strategies

Table 1 Algorithmic variants of the policy gradient algorithm

Algorithm	Reward signal	Description
Policy gradient	$\hat{v}_{n,j}^m$	Cumulative reward
Policy gradient with baseline	$\hat{v}_{n,j}^m - \bar{v}_{j_\tau}^m$	Deduct average reward
Q-value	$Q(b_{n,j}^m, j, J_n)$	Estimated cumulative reward
TD(1)	$\hat{v}_{n,j}^m - Q(b_{n,j}^m, j, J_n)$	Deduct Q-value
Advantage value	$Q(b_{n,j}^m, j, J_n) - Q(\mu_\theta, j, J_n)$	Deduct average Q-value

4.2 Policy gradient extensions

Section 4.1 presented the basic policy gradient algorithm. We now introduce four extensions, namely (i) policy gradient with baseline, (ii) actor-critic with Q-value, (iii) temporal difference learning—more specifically TD(1)—and (iv) actor-critic with advantage function. The algorithms are summarized in Table 1; for detailed descriptions we refer to Sutton and Barto (2018).

We first discuss the policy gradient with baseline. Rewards may exhibit large variance that hampers learning. To reduce this variance, we deduct the average observed value $\bar{v}_{j_\tau}^m$ during the episode as a baseline value (Sutton and Barto 2018). In our update procedure, we then replace $\hat{v}_{n,j}^m$ with $\hat{v}_{n,j}^m - \bar{v}_{j_\tau}^m$, yielding lower variance than the original.

The second extension is the actor-critic algorithm with Q-values; a hybrid of policy approximation and value approximation. Figure 2 provides an illustration of an actor-critic architecture. Policy gradient methods rely on directly observed rewards, which may strongly vary between episodes. Furthermore, actor networks do not leverage information about particular states encountered. In the actor-critic approach, we replace the observed value $\hat{v}$ with a function $Q(b_{n,j}^m, j, J_n)$ that is often defined by a neural network (critic network). The critic network transforms the input features into an expected reward value, popularly known as a Q-value. Drawbacks of actor-critic methods are (i) the need to learn additional parameters, (ii) slower convergence than actor-only methods and (iii) simultaneous adjustments of strategy and value function. Particularly the latter is problematic in highly non-stationary multi-agent settings; value functions learned in the past may no longer be representative for current strategies and vice versa.

The third extension, temporal difference learning, utilizes the Q-value as a baseline by subtracting it from the observed rewards. We use the TD(1) variant that incorporates the full reward trajectory (Sutton and Barto 2018). Unlike TD(0), the reward signals are unbiased. Subtracting the Q-value yields a reward signal $\hat{v}_{n,j}^m - Q(b_{n,j}^m, j, J_n)$. This approach preserves both the actual observations and the value function approximations while reducing variance.

Fig. 2 Example of actor network (left) outputting μ and σ and critic network (right) outputting $\bar{V}$. The input layer contains the feature vector ϕ

The fourth extension we discuss is the advantage function (also known as Advantage Actor Critic or A2C). It also uses a baseline function, but utilizes value functions rather than reward observations. Specifically, we define the reward signal $Q(b_{n,j}^m, \mathbf{j}, \mathbf{J}_n) - Q(\mu_\theta, \mathbf{j}, \mathbf{J}_n)$, where the baseline term is a reward function that depends on the state but is independent of the action (bid or ask). Concretely, the first Q-value uses the sampled bid/ask as a feature, the second Q-value the mean bid/ask. Again, the objectives are to reduce the variance and to generalize past observations.

4.3 Update procedure

Algorithm 2 outlined the generic policy gradient algorithm with a generic update function $U(\theta^{S,m}, \mathbf{I}^{S,m})$. Here we discuss the updating procedure in more detail. Traditionally, policy gradient algorithms follow stochastic gradient ascent for updates. The corresponding gradients can be computed with respect to each feature and are defined by

$$\begin{aligned}\nabla_{\mu_\theta}(\mathbf{j}, \mathbf{I}^m) &= \frac{(b_{n,j}^m - \mu_\theta(\mathbf{j}, \mathbf{J}_n))\phi(\mathbf{j}, \mathbf{J}_n)}{\sigma_\theta^2}, \\ \nabla_{\sigma_\theta}(\mathbf{j}, \mathbf{I}^m) &= \frac{(b_{n,j}^m - \mu_\theta(\mathbf{j}, \mathbf{J}_n))^2 - \sigma_\theta^2}{\sigma_\theta^3}.\end{aligned}$$

In a neural network setting, we might compute these gradients with respect to the activation functions and determine the corresponding updates for the network weights (classical stochastic gradient ascent). However, it is often convenient to define a loss function that allows updating with state-of-the-art gradient *descent* algorithms, using a backpropagation procedure. Update algorithms such as ADAM often outperform traditional gradient descent. The Gaussian loss function (Van Heeswijk et al. 2019b) is defined by:

$$\mathcal{L}^{actor}(b_{n,j}^m, \mathbf{J}_n, \hat{v}_j) = -\log \left(\frac{1}{\sqrt{2\pi}\sigma_\theta} e^{-\frac{(b_{n,j}^m - \mu_\theta(\mathbf{j}, \mathbf{J}_n))^2}{2\sigma_\theta^2}} \right) \hat{v}_j. \quad (8)$$

To update critic networks, we also start with a loss function such that we can perform backpropagation. This loss function is simply the mean-squared error between Q-value and observed rewards:

Table 2 Settings for Case I (deterministic) and Case II (stochastic)

Setting	Case I	Case II
Number of job arrivals	1	[1–10]
Due date	0	[1–5]
Job transport distance	1	[1–5]
Job volume	1	[1–5]
Willingness to pay (per volume unit per mile)	2	2
Transport costs (per volume unit per mile)	1	1
Transport capacity	1	{40, 300}

$$\mathcal{L}^{critic}(b_{n,j}^m, J_n, \hat{v}_j) = (\hat{v}_{j\tau,j}^m - Q(b_{n,j}^m, j, J_n))^2. \quad (9)$$

5 Experimental design

This section introduces the experimental design. The main objective is to provide insights into the behavior of the algorithm. Section 5.1 introduces the two test cases and Sect. 5.2 defines the performance metrics. Section 5.3 describes the Python implementation. Finally, Sect. 5.4 lists the features used as network input.

5.1 Case properties

We present two test cases for this study; a toy-sized deterministic one (Case I) and a larger stochastic one (Case II).

In Case I (deterministic), each day exactly one job arrives with $j_\tau = 0$ (i.e., only one decision epoch), volume $j_v = \zeta^C$ and a fixed distance j_d . It follows that the job is shipped if $a_{n,j}^m < b_{n,j}^m$ and fails otherwise. The maximum willingness to pay is 2, the transport costs are 1. This simple setting allows to test behavior in-depth under a variety of circumstances. In addition, it abstractly links to a setting in which modular containers bid in unison (e.g., as a horizontal collaboration or cluster of containers) (Sallez et al. 2016). We use Case I to tune parameters, explore the parameter space and obtain behavioral insights.

Case II stochastically generates a number of job arrivals each day and includes varying job properties. Due dates, volumes and distances range between 1 and 5. Per volume unit per mile, the maximum willingness to pay is 2 and transport costs are 1. The number of jobs arriving daily varies between 0 and 10; the total number of jobs may accumulate up to 50. We consider two variants of the case, one in which the transport capacity is 40 (somewhat scarce) and one where it is 300 (abundant). The additional challenges in Case II, compared to Case I, stem from the uncertain availability of sufficient capacity and the varying dimensions of the jobs. Table 2 summarizes the case settings.

5.2 Performance metrics

The objective of our market design is to represent a completely decentralized, self-organizing transport market without interventions or regulations. The performance metrics are aligned

to this purpose. Remind that we study an online environment, subject to strategy updates while measuring performance.

In Sect. 3.5, we established that infinitely many Nash equilibria exist for our system. However, the Nash equilibrium is a theoretical result that emerges in the limit after all uncertainty has been resolved. The setting studied here is inherently uncertain; due to the stochastic nature of the policies perfect adherence is never achieved. Therefore, adherence to the Nash equilibrium is the first performance metric. Formally, we define Nash adherence as follows:

$$\max \left(0, \sum_{t \in \mathcal{T}_j} \gamma_{t,j} \frac{(a_{n,j}^m - c_j^{min}) + (c_j^{max} - b_{n,j}^m)}{(c_j^{max} - c_j^{min})} \right).$$

The second performance metric is fairness. Although a Nash equilibrium is rational, it is not necessarily perceived as fair; one agent might receive the full system gain. Section 3.5 introduced several normative solutions that guarantee Pareto optimality and symmetry of utilities, setting an upper bound for fairness. For our linear utility functions, the solutions boil down to a 50/50 split. We note that agents compete to increase their own reward share in a stochastic setting with incomplete information. As such, unfairness is an inherent aspect of competitive markets and not necessarily an indication of poor performance. We formalize fairness with the following metric:

$$\max \left(0, 1 - \left| \sum_{t \in \mathcal{T}_j} \gamma_{t,j} \frac{(a_{n,j}^m - c_j^{min}) - (c_j^{max} - b_{n,j}^m)}{(a_{n,j}^m - c_j^{min}) + (c_j^{max} - b_{n,j}^m)} \right| \right).$$

The third metric is utilization. If a transport service departs with idle capacity while feasible jobs (in terms of capacity) remain unshipped, this indicates an inefficiency in the market. Also note that the preceding metrics suffer due to low utilization. To determine the upper bound of utilized capacity, we solve a variant of the knapsack problem in which we substitute the job value with the job volume, thus maximizing utilization of transport capacity:

$$\arg \max_{\gamma_j^{\zeta^C} \in \Gamma^{\zeta^C}(J_n)} \left(\sum_{j \in J_n} \gamma_j^{\zeta^C} \cdot j_v \right), \quad (10)$$

s.t.

$$\sum_{j \in J_n} \gamma_j^{\zeta^C} \cdot j_v \leq \zeta^C.$$

Dividing the true utilization by the theoretically possible utilization gives the relative utilization:

$$\frac{\sum_{j \in J_n} \gamma_{n,j} \cdot j_v}{\sum_{j \in J_n} \gamma_{n,j}^{\zeta^C} \cdot j_v} \quad (11)$$

As a concluding remark, we emphasize that the learning algorithms are purposefully designed to be model-free, and the policies to be stochastic. This way, we make minimal assumptions on the behavior of agents. We admit, however, that deterministic or rule-based policies likely achieve better system-wide performance. The present study's objective is not to achieve the best possible performance (as the solution itself is trivial), but instead observe behavior under minimalistic behavioral assumptions.

5.3 Implementation

The algorithm is coded in Python 3.8 and uses TensorFlow 2.3 for the neural network architectures. To initialize the neural network weights, we use He initialization (He et al. 2015) for all but the last layer. For the final layer of the actor networks, we initialize bias weights such that the carrier's ask equals $c^{C, trn}$ and the shipper's bid equals $c^{S, max}$ (we test different bias initializations in later experiments). For the Q-value (critic network), all weights in the final layer are initialized at 0, guaranteeing an unbiased initial estimate that does not influence decisions in the early iterations. We use the ADAM optimizer for network updates, which we find to consistently outperform optimizers such as classic stochastic gradient descent and RMSprop.

All solution methods are verified by checking the behavior of trivial strategies (e.g., by fixing the strategy of the opposing agent). We remark that our actor-critic methods (requiring to learn a Q-value) generally take long to converge even in simple settings, but their eventual convergence is confirmed.

We perform 5 replications for the main experiments (after the initial explorations), which in stable experiments gives standard deviations less than 5% on the KPIs. We report average values (excluding the first 10% of iterations, which serve as a warm-up period) and end-of-horizon results, the latter provide convergence insights.

5.4 Features

To parametrize the strategy we use several features, denoted by vector $\phi(j, J_n)$. This vector (unique for each job) serves as input layer for the neural networks. We adopt the same set of features as defined in Van Heeswijk (2019) for the single-agent system, namely (i) bias, (ii) job due date, (iii) job transport distance, (iv) job volume, (v) average # epochs till due date, (vi) average distance, (vii) average volume and (viii) total volume. Note that the features contain both individual job- and system-wide properties, implying that the containers share information (but not between agents). Such information-sharing is shown to have a small but positive impact on solution quality (Van Heeswijk 2019). The proposed feature set demonstrates expected patterns, such as bid prices increasing closer to the due date and higher bids being placed for larger volumes.

For computing Q-values, we add the selected bid- or ask price to the vector; for the advantage function we use the average bid/ask price. "Appendix F" mathematically defines the features. By initializing weights corresponding to the bias, we determine the initial values for μ_θ^0 and σ_θ^0 . In the implementation, all features are normalized before being used as input for the neural networks.

6 Numerical results and analysis

This section presents the results and analysis of the numerical experiments, as outlined in Sect. 5. In Sect. 6.1 we explore the parameter space and determine suitable hyper-parameters (based on Case I). Section 6.2 analyzes behavior for the deterministic Case I, Sect. 6.3 for the stochastic Case II.

Fig. 3 Convergence with transport costs 1 and maximum willingness 2

6.1 Exploration of parametric space

The experiments in this section (utilizing Case I) set suitable hyper-parameters for the subsequent experiments and provide insights into various architectural settings. After validating basic mechanisms, the aspects we test are (i) number and length of episodes, (ii) learning rates, (iii) initial standard deviations, (iv) actor network architectures, (v) actor-critic variants and (vi) critic network architectures. In the result tables, we provide average results (excluding the 10% warm-up period) and end-of-horizon results between parentheses.

6.1.1 Verification

We highlight a few verification experiments to demonstrate algorithmic behavior, using simplified system settings. If we fix a single agent's bid (ask), the other agent's ask (bid) converges to the fixed value when within the feasible region. If both agent adopt equivalent strategies, we find that bid- and ask prices gravitate roughly towards the average of maximum willingness and transport costs, i.e., a fair balance. Figures 3, 4, 5 and 6 illustrate some of the initial (pre-finetuning) behaviors observed during verification.

6.1.2 Episode lengths

We test the effects of adjusting the number and length of episodes. Smaller batches of observations result in more frequent but potentially less stable strategy updates. We fix the total number of simulated days at 1 million, selecting $m \in \{10, 100, 1000, 10,000\}$ episodes with the appropriate number of corresponding days per episode. With 1000 or fewer strategy updates, the solution quality notably decreases (Table 3). We use 1000 episodes of length 1000—yielding both adherence and fairness over 0.90—in the remaining experiments.

Fig. 4 Convergence with transport costs 1.5 and maximum willingness 3

Fig. 5 Convergence with bid fixed at 2.0

6.1.3 Learning rates

We test a variety of learning rates $\alpha \in \{0.1, 0.01, 0.001, 0.0001, 0.00001\}$. Higher learning rates are more responsive to changes in the system, but also display worse convergence behavior (Table 4). Learning rates 0.1 and 0.01 are unstable due to exploding gradients. We find that 0.001 generally provides the best average results, but lower learning rates yield better results after 100,000 iterations. As we perform online learning that requires both responsiveness and quality, we set $\alpha = 0.001$.

Fig. 6 Convergence with ask fixed at 1.0

Table 3 Comparison for various numbers of batches and episode lengths m

# Batches	m	Utilization	Nash adherence	Fairness
100,000	10	1.00 (1.00)	0.95 (0.96)	0.88 (0.91)
10,000	100	0.99 (0.99)	0.93 (0.93)	0.90 (0.85)
1000	1000	0.99 (0.99)	0.92 (0.94)	0.93 (0.92)
100	10,000	0.89 (0.99)	0.78 (0.87)	0.82 (0.92)
10	100,000	0.71 (0.90)	0.51 (0.66)	0.65 (0.83)

A balance between yields both adherence and fairness over 0.90 on average

Table 4 Comparison for various learning rates

α	Utilization	Nash adherence	Fairness
0.1	N/A	N/A	N/A
0.01	N/A	N/A	N/A
0.001	0.99 (0.99)	0.92 (0.94)	0.92 (0.92)
0.0001	0.99 (1.00)	0.94 (0.97)	0.73 (0.96)
0.00001	0.94 (1.00)	0.78 (0.98)	0.78 (0.93)

Low learning rates display better convergence, but worse results on average

6.1.4 Initial standard deviations

We test initial standard deviations σ_θ^0 ranging from 0.01 to 2 (results in Table 5). Overall, results are quite close. Lower initial standard deviations encourage less exploration and increase Suggestions for editing time to converge, whereas a larger deviation may cause stronger fluctuations that impact fairness in particular. A standard deviation of 0.1 yields the best results for this particular setting. We note that—regardless of initialization—standard deviations converge to similar, relatively low values eventually. However, there seemingly is a

Table 5 Comparison for standard deviation initializations

σ_{θ}^0	Utilization	Nash adherence	Fairness
0.01	0.99 (0.99)	0.89 (0.94)	0.86 (0.88)
0.1	0.99 (0.99)	0.91 (0.93)	0.92 (0.93)
0.5	0.99 (0.99)	0.91 (0.86)	0.86 (0.82)
1.0	0.99 (1.00)	0.91 (0.91)	0.95 (0.95)
1.5	0.99 (0.99)	0.90 (0.92)	0.89 (0.86)
2.0	0.99 (0.99)	0.91 (0.91)	0.85 (0.94)

Results are close, but $\sigma_{\theta}^0 = 0.1$ strikes a good balance between fairness and Nash adherence

Table 6 Comparison of various actor network architectures

Architecture	Utilization	Nash adherence	Fairness
Linear	0.99 (1.00)	0.96 (0.97)	0.86 (0.97)
1L 20N	0.99 (1.00)	0.91 (0.91)	0.95 (0.95)
2L 20N	0.99 (0.99)	0.89 (0.91)	0.92 (0.94)
3L 20N	0.99 (0.99)	0.89 (0.93)	0.92 (0.96)

The 1-layer network with 20 nodes yields the best performance on average. The linear architecture performs best on Nash adherence but worst on fairness

benefit to preserve some randomness in the actions, as standard deviations do not completely converge to 0. This way, opponents can never perfectly counter the strategy.

6.1.5 Actor networks architectures

We now address actor network architectures. We test a setting without hidden layer (i.e., a linear model) and 1–3 layers with 5–30 nodes each. The input dimension equals the number of features $|\theta|$, the output contains two nodes (μ_{θ} and σ_{θ}). Adding hidden nodes and -layers may capture more complicated functions, but take both additional time and observations to train. Table 6 shows the key results; the full table of experiments is displayed in “Appendix B”. First, we find that performance is fairly stable across architectures. Some sub-par convergence behavior is noted for both the smallest (5 nodes) and largest (25 or 30 nodes) networks, although all converge to good (0.90+) metric scores eventually. Second, computational times are fairly similar for various network sizes, with 10% difference between network configurations. Third, the linear approximation schemes outperforms the neural network architectures with respect to eventual performance, but does notably worse on average fairness. We continue with an actor network consisting of one layer with 20 nodes. As Rolnick and Tegmark (2017) point out, relatively small neural networks suffice for many real-world problems. For our system, this notion appears to apply as well.

6.1.6 Policy gradient algorithms

We test five variants of the policy gradient algorithm: (i) vanilla policy gradient, (ii) policy gradient with baseline, (iii) actor-critic with Q-value, (iv) TD(1) and (v) actor-critic advantage function. The critic- and actor networks use the same architecture; one layer with 20 nodes. The key results are summarized in Table 7. The policy gradient algorithms consistently

Table 7 Comparison of policy gradient algorithms

Algorithm	Utilization	Nash adherence	Fairness
Policy gradient	0.99 (0.99)	0.92 (0.94)	0.93 (0.92)
Policy gradient w. baseline	0.99 (0.99)	0.90 (0.91)	0.95 (0.96)
Q-value	0.82 (0.98)	0.69 (0.86)	0.66 (0.81)
TD(1)	0.09 (0.00)	0.07 (0.00)	0.08 (0.00)
Advantage value	0.98 (1.00)	0.00 (0.00)	0.00 (0.00)

The actor-critic algorithms suffer from poor convergence

Table 8 Comparison of critic network architectures

Architecture	Utilization	Nash adherence	Fairness
Q-value linear	0.95 (0.95)	0.05 (0.17)	0.00 (0.00)
Q-value 1L20N	0.82 (0.98)	0.69 (0.86)	0.66 (0.81)
Q-value 2L20N	0.06 (0.57)	0.05 (0.51)	0.00 (0.00)
Q-value 3L20N	0.17 (0.91)	0.13 (0.77)	0.57 (0.30)

Similar to the actor networks, a 1-layer network with 20 nodes yield the best results. Overall, performance when adding a critic network is subpar

outperform the actor-critic methods; especially TD(1) and Advantage Value suffer from premature convergence and perform very poorly. As target values keep changing over time, simultaneously updating the strategies and value functions is notoriously hard. An incorrect Q-value leads to poor strategies and vice versa, as reflected in the performance. This observation is in line with Grondman et al. (2012), stating that actor-critic methods are less suitable for highly non-stationary environments. We therefore prefer policy gradient algorithms; the vanilla- and baseline variants perform similarly.

6.1.7 Critic network

Finally, we test various architectures for the critic network, using either a linear approximation scheme or 1–3 hidden layers. The one-layer network outperforms all alternatives (Table 8). Furthermore, it is confirmed again that the actor-critic approaches do not achieve consistent results.

6.1.8 Summary exploration

We summarize the main conclusions of our exploration. Due to the highly non-stationary environment, convergence to stable solutions is relatively slow. Policy gradient algorithms without a critic deal much better with this non-stationarity, responding more quickly to environmental changes. With the exception of critic-based solutions, we find solutions to be fairly stable across test settings. Still, there is an inherent trade-off between responsiveness and solution quality. Deep neural networks and low learning rates yield good results for static targets (reflected in the end-of-horizon results), but average performance worsens in dynamic (e.g., online) settings. For our remaining experiments, we therefore adopt an equal balance between batch size and number of episodes (both 1000), a learning rate of 0.001, an

Table 9 Comparison of asymmetric learning rates

Learning rate (S)	Utilization	Nash adherence	Reward share (S)	Reward share (C)
0.0001	0.99 (1.00)	0.92 (0.95)	0.33 (0.40)	0.51 (0.56)
0.0005	0.99 (0.99)	0.87 (0.91)	0.37 (0.45)	0.47 (0.50)
0.001	0.99 (0.99)	0.92 (0.93)	0.40 (0.43)	0.43 (0.48)
0.002	0.99 (0.99)	0.91 (0.92)	0.43 (0.44)	0.40 (0.48)
0.005	0.99 (0.99)	0.91 (0.94)	0.43 (0.48)	0.39 (0.47)
0.01	N/A	N/A	N/A	N/A

The carrier has a fixed learning rate of 0.001. Higher learning rates yield an advantage in terms of reward share

initial standard deviation of 0.1, and an actor-network consisting of one hidden layer with 20 neurons.

6.2 Analysis case I

In Sect. 6.1 we discussed the elementary system behavior and determined appropriate parameters. We now further delve into the competitive elements of the environment, with both agents aiming to obtain the best deal for themselves by making strategic decisions. In addition to the system metrics, we therefore also consider average rewards as agent-specific metrics. We present the reward shares for carrier and shipper instead of fairness; remind that the remaining share goes to the broker.

We explore whether agents can outperform their counterpart with a deviating strategy. For convenience we mostly keep the carrier's behavior fixed and alter that of the shipper; similar results are obtained the other way around. This section addresses the following aspects: (i) asymmetric learning rates, (ii) penalty function, (iii) actor network architecture, (iv) actor-critic algorithm, (v) bias initialization and (vi) standard deviation initialization. Based on the obtained insights, we then run a number of experiments in which both shipper's and carrier's strategies vary based on their risk profile, trying to determine the best response to the opponent's strategy.

6.2.1 Asymmetric learning rates

The learning rate determines how responsive the strategy updates are to new information. If an agent uses a higher learning rate than its opponent, it could adapt faster to a new environment. We find that, to a certain degree, higher learning rates yields better results (Table 9), with the shipper outearning the carrier by up to 9%. For learning rates of 0.01 and higher, we did not consistently find stable solutions.

6.2.2 Penalty function

The penalty functions' slopes [Eqs. (1)–(2)] determine the artificial penalty incurred when jobs are not shipped. By default both have a slope of 1, such that missed rewards are weighted the same as realized rewards. Here we test a number of alternative slopes, ranging from 0 (i.e., no penalty) to 5 (high risk aversion). We find that lower penalties average yield strategies with higher realized rewards (Table 10). A shipper without a penalty function obtains up to 68% of the revenues, versus only 26% for the carrier (the remainder going to the broker).

Table 10 Comparison of asymmetric penalty functions

Penalty slope (S)	Utilization	Nash adherence	Reward share (S)	Reward share (C)
0	0.98 (0.98)	0.92 (0.93)	0.64 (0.68)	0.21 (0.26)
0.5	0.99 (0.99)	0.93 (0.94)	0.49 (0.51)	0.34 (0.42)
1.0	0.99 (0.99)	0.92 (0.93)	0.40 (0.43)	0.43 (0.48)
2.0	0.99 (0.99)	0.93 (0.94)	0.32 (0.38)	0.51 (0.56)
5.0	0.99 (1.00)	0.93 (0.94)	0.26 (0.33)	0.58 (0.61)

The carrier has a fixed penalty slope of 1. Lower penalties considerably benefit the shipper

Nash adherence and utilization are only marginally affected, but do go down slightly with risk-seeking behavior.

6.2.3 Linear approximation versus actor network

By default we use actor networks with 1 layer and 20 neurons. We now test whether adopting a linear approximation scheme by the shipper yields a benefit or disadvantage, being more responsive but less powerful. We find that, in this setting, the average reward shares are 32% (was 40%) for the shipper and 53% (was 43%) for the carrier. This substantial difference implies that the neural network improves decisions.

6.2.4 Q-value versus policy gradient

Section 6.1 already indicated that algorithms relying on value function approximation perform much worse than those relying on reward observations. We now test a setting in which only the shipper uses an actor-critic method, checking once more whether the additional expressive power of the critic network could give an advantage. Again, the results are poor. Nash adherence is only 65%, and on average the shipper earns a negative reward. Although some runs yield reasonable outcomes for the shipper, others yield unstable solutions in which shipping agreements rarely materialize.

6.2.5 Initial bias

In Sect. 6.1 we assumed that an agent's initial bid (ask) equals its willingness to pay (marginal transport costs). However, this directly reveals their highest bid; in reality they may start with a lower bid (higher ask) and update until finding a balance. Thus, we conduct an experiment in which the shipper starts with lower bids; the initial average bid is set at 0, 0.5, 1.0, 1.5, or 2.0. Naturally, the bias weight is updated over time as before. The results (Table 11) show that starting out with a low bid is clearly beneficial, with the bid-ask pairs converging to equilibria that are considerably better for the shipper.

6.2.6 Initial standard deviation

A higher initial standard deviation than the opponent enables exploring more, yet also increases the risk of ending up in unfeasible solution regions. We test for $\sigma_{\theta}^{S,0} \in \{0.01, 0.1, 0.5, 1, 1.5, 2.0\}$. This only entails the initial standard deviations; we find them

Table 11 Comparison of varying initial bias

Bias $\mu_{\theta}^{S,0}$	Utilization	Nash adherence	Reward share shipper	Reward share carrier
0.0	0.99 (0.99)	0.93 (0.92)	0.81 (0.62)	0.23 (0.32)
0.5	0.99 (0.99)	0.93 (0.93)	0.75 (0.60)	0.25 (0.34)
1.0	0.99 (0.99)	0.94 (0.95)	0.66 (0.62)	0.29 (0.33)
1.5	0.99 (0.99)	0.93 (0.93)	0.51 (0.49)	0.39 (0.45)
2.0	0.99 (0.99)	0.92 (0.93)	0.40 (0.43)	0.43 (0.48)

The carrier's initial bias is set at $\mu_{\theta}^{C,0} = 1$. Low bias initializations strongly improve performance for the shipper

Table 12 Comparison of varying initial SD

SD $\sigma_{\theta}^{S,0}$	Utilization	Nash adherence	Reward share shipper	Reward share carrier
0.01	0.99 (0.99)	0.93 (0.93)	0.32 (0.44)	0.52 (0.50)
0.1	0.99 (0.99)	0.93 (0.94)	0.36 (0.46)	0.49 (0.49)
0.5	0.99 (0.99)	0.92 (0.93)	0.40 (0.47)	0.44 (0.47)
1.0	0.99 (0.99)	0.92 (0.93)	0.40 (0.43)	0.43 (0.48)
1.5	0.99 (0.99)	0.92 (0.93)	0.38 (0.42)	0.45 (0.51)
2.0	0.99 (0.99)	0.92 (0.92)	0.35 (0.41)	0.49 (0.52)

The carrier's initial st. dev. is set at $\sigma_{\theta}^{C,0} = 1$. Middle-range exploration yields better results for the shipper

all converging to similar values regardless of initialization. The experiments show a clear disadvantage when the shipper uses lower standard deviations (Table 12). For higher standard deviations performance also decreases, although not as sharply. There appears no imminent benefit to alter the standard deviation, as the default value yields the best outcome.

6.2.7 Shipper versus carrier

After completing the experiments varying only the shipper's behavior, we design experiments in which both agents adopt strategy settings reflecting a certain risk appetite. We define three profiles—risk-seeking, risk-neutral, risk-averse—varying in initial bias, penalty slope and learning rate (see Table 13).

Using the risk profiles, we perform nine experiments pitting each profile pair. The results are shown in Table 14. Overall, the more risk-seeking agent consistently obtains better rewards. Figures 7, 8, 9 and 10 illustrate the dynamics between agents with different risk profiles for four settings. The bid-ask prices converge to equilibria that favor the agent with higher risk tolerance. However, when both agents use a risk-seeking profile, they stick to their initial bias and never explore feasible solution regions, partially due to the absence of penalties. This implies that reward shaping is desirable for a functioning system. We also tested variants with two risk-seeking agents (e.g., with small penalty slopes or larger standard deviations), but they tend to be unstable.

In terms of system performance, we find no major differences, with utilizations close to 1 and Nash adherence generally between 0.90 and 0.95. Occasionally, systems with a systems with a risk-seeking agent fail to find equilibria, (temporarily) resulting in non-functioning transport markets. From a game-theoretical perspective, “Appendix D” shows that when

Table 13 Risk profiles for carrier and shipper

Risk profile	Carrier	Shipper
Risk-seeking	Init. Bias = 1	Init. Bias = 2
	Penalty slope = 2	Penalty slope = 2
	Learning rate = 0.005	Learning rate = 0.005
Risk-neutral	Init. Bias = 1	Init. Bias = 2
	Penalty slope = 1	Penalty slope = 1
	Learning rate = 0.001	Learning rate = 0.001
Risk-averse	Init. Bias = 1	Init. Bias = 2
	Penalty slope = 2	Penalty slope = 2
	Learning rate = 0.0001	Learning rate = 0.0001

A risk-averse profile is aimed at finding feasible solutions, a risk-seeking profile at maximizing individual gains

Table 14 Performance for various risk profiles

Carrier	Shipper		
	RS	RN	RA
RS	N/A [0.00]	0.81/0.18 [0.85]	0.82/0.14 [0.95]
RN	0.25/0.75 [0.97]	0.46/0.49 [0.99]	0.62/0.28 [0.99]
RA	0.11/0.85 [0.98]	0.26/0.63 [0.99]	0.41/0.40 [1.00]

Each cell reports 'reward share carrier/reward share shipper [utilization]'. Risk-seeking behavior if generally rewarded, but not if both agents engage in it

the opponent is risk-seeking, the best response is to be either risk-averse (for carrier) or risk-neutral (for shipper).

6.2.8 Managerial insights case I experiments

From the experiments, we draw a number of conclusions. We reconfirm that actor-critic methods yield poor performance and that actor networks outperform linear approximations. Furthermore, risk-seeking behavior generally pays off. By responding faster to market fluctuations (higher learning rate), placing lower penalty weights on failed shipments, and setting a bold initial opening bid/ask, a competing agent may be outperformed. However, we also find evidence of destabilizing effects and poor market efficiency due to risk-seeking behavior. Without central regulation or communication between agents, risk-seeking behavior might have grave consequences; game theory may guide intelligent responses to the other agent's strategy. As performance differences can be quite substantial, these findings imply that algorithmic optimization is crucial to successfully participate in decentralized freight transport markets.

6.3 Analysis case II

This section analyzes the second experimental setting, which generates up to 10 jobs per day with varying properties in terms of volume, due date and distance. Case II serves as a proof-of-

Fig. 7 Carrier RA and shipper RA

Fig. 8 Carrier RN and shipper RN

concept and to measure performances in a more realistic setting. To recap: we now consider 125 distinct job types (varying in due date, volume and distance), with order accumulation possible up to 60 jobs. With a maximum daily volume of 50 and a transport capacity of either 40 (somewhat scarce) or 300 (abundant), we evaluate instances that favor either carrier or shipper. We re-use the same network parameters as fine-tuned for Case I. Proper re-tuning on Case II might elevate performance, however, preliminary experiments gave insufficient indication that results would improve consistently and significantly. Following Sect. 6.2.7, we again utilize the three risk profiles, starting by using the same profiles for both agents. The bias initialization procedure is comparable to Case I, but notationally more involved and therefore formalized in “Appendix C”.

The initial results of pairing the three risk profiles illustrate the increased challenges of the stochastic case (Table 15). The risk-seeking and risk-neutral profiles do not consistently yield

Fig. 9 Carrier RS and shipper RS

Fig. 10 Carrier RA and shipper RS

stable solutions. For the risk-averse profiles, utilization ranges from 67 to 89%, indicating an increased difficulty to find suitable transport matches. Furthermore, adherence to the Nash equilibrium drops to 65%, revealing substantial market inefficiencies. Bid-ask spreads are higher due to the large uncertainty in the market; the broker now takes 35% of the market value. Finally, fairness and reward shares indicate unbalances in the market, with a tendency to favor the carrier. Recall from Eqs. (1)–(2) that penalty functions are not symmetric; omitting the idle capacity constraint for the carrier yields fair (but less sensible) outcomes.

To improve system performance, we execute a number of follow-up experiments. Using the insights obtained from Case I experiments, we vary the initial bias and sigma, the learning rate, and the penalty slope. As only risk-averse profiles yield consistent results, we consider several variants of that profile. Table 16 shows that most modifications are unsuccessful in

Table 15 Results Case II per risk profile, with both agents using the same profile

Risk profile	Case variant (capacity)	Utilization	Nash adherence	Fairness	Reward share shipper	Reward share carrier
Risk-seeking	$\zeta^C = 40$	N/A	N/A	N/A	N/A	N/A
Risk-seeking	$\zeta^C = 300$	N/A	N/A	N/A	N/A	N/A
Risk-neutral	$\zeta^C = 40$	0.86 (0.86)	0.53 (0.43)	0.14 (0.16)	0.34 (0.01)	0.23 (0.20)
Risk-neutral	$\zeta^C = 300$	N/A	N/A	N/A	N/A	N/A
Risk-averse	$\zeta^C = 40$	0.89 (0.80)	0.65 (0.67)	0.13 (0.14)	0.05 (0.13)	0.74 (0.73)
Risk-averse	$\zeta^C = 300$	0.67 (0.76)	0.65 (0.22)	0.23 (0.04)	0.06 (0.12)	0.73 (0.73)

Compared to Case I, Nash adherence drops substantially. Risk-seeking profiles do not achieve stable results

Fig. 11 Carrier RA bias and shipper RA bias ($\zeta^C = 40$)

achieving notable improvements, yet using a risk-neutral bias initialization (i.e., halfway expected willingness to pay and expected transport costs) strongly improves performance. We find a utilization of 0.98–0.99 and a Nash adherence of 0.84–0.87. Although fairness remains low (recall that fairness is calculated at the individual job level), for overall profits the shares of carrier and shipper differences are less pronounced (0.54/0.39 and 0.60/0.32).

The system effectiveness of starting with a fair bid/ask is an important result, yet not necessarily individual rational. The shipper's performance improves drastically (the reward share increases from 0.05 to 0.54), but the carrier's reward share drops from 0.74 to 0.39. The increased number of jobs shipped does not compensate for that. As a final experiment, we therefore again test the impact of asymmetric profiles, determining individually rational bias initializations. Although we describe biases in terms of risk, the learning rate and penalty function remain risk-averse. The results are shown in Table 17 ($\zeta^C = 40$) and Table 18 ($\zeta^C = 300$). As before, risk-seeking approaches generally improve individual performance, although the results are less congruous and clear-cut than for the deterministic case.

We discuss best responses of individual agents, considering bias initialization as a game on itself. Formally, we achieve Nash equilibria² when the shipper adopts the risk-seeking bias initialization and the carrier uses a risk-neutral bias ($\zeta^C = 40$) or risk-averse bias ($\zeta^C = 300$), see “Appendix D” for details and payoff matrices. With a somewhat less rigorous interpretation, the payoffs imply that if one agent adopts a bold bias, the opponent should settle for lower individual gains. If both agents use bold bid/ask strategies, systems do not converge properly and individual gains are minimal as a result. Figures 11, 12, 13 and 14 highlight results for four risk profile pairs, including the two Nash equilibria.

To conclude, in the stochastic case (with 125 job types and uncertain numbers of jobs) stability is harder to achieve than in the deterministic case, requiring more cautious updating strategies. The results suggest that clustering containers for joint bids (as is implicitly done in Case I) yields more stability. The main determinant for the agents' individual gains is

² The Nash equilibrium discussed here (for the game of initializing the bias) should not be convoluted with the Nash equilibrium for the bargaining game as mentioned earlier.

Table 16 Results Case II for variants of risk-averse profiles

Parameter setting	Case variant (capacity)	Utilization	Nash adherence	Fairness	Reward share shipper	Reward share carrier
$lr=0.00001$	$\zeta^C = 40$	0.98(0.98)	0.41 (0.61)	0.09 (0.13)	0.06 (0.03)	0.43 (0.76)
$lr=0.00001$	$\zeta^C = 300$	0.98(0.98)	0.42 (0.59)	0.09 (0.13)	0.05 (0.00)	0.46 (0.77)
$\sigma = 10$	$\zeta^C = 40$	0.99 (1.00)	0.53 (0.54)	0.13 (0.12)	0.30 (0.38)	0.36 (0.32)
$\sigma = 10$	$\zeta^C = 300$	0.99 (0.99)	0.53 (0.60)	0.12 (0.11)	0.31 (0.42)	0.37 (0.34)
Pen. slope = 0	$\zeta^C = 40$	0.01 (0.00)	0.01 (0.00)	N/A	N/A	N/A
Pen. slope = 0	$\zeta^C = 300$	0.13 (0.10)	0.12 (0.13)	0.22 (0.23)	0.87 (N/A)	-0.02 (N/A)
Pen. slope = 1	$\zeta^C = 40$	0.97 (0.95)	0.62 (0.69)	0.13 (0.14)	-0.03 (-0.08)	0.81 (0.93)
Pen. slope = 1	$\zeta^C = 300$	0.99 (1.00)	0.60 (0.64)	0.13 (0.14)	-0.26 (-0.76)	0.78 (0.89)
RN bias	$\zeta^C = 40$	0.99 (0.99)	0.87 (0.91)	0.14 (0.16)	0.54 (0.56)	0.39 (0.40)
RN bias	$\zeta^C = 300$	0.98 (0.98)	0.84 (0.89)	0.14 (0.15)	0.60 (0.63)	0.32 (0.31)
RS bias	$\zeta^C = 40$	N/A	N/A	N/A	N/A	N/A
RS bias	$\zeta^C = 300$	N/A	N/A	N/A	N/A	N/A

Setting a risk-neutral bias has by far the largest positive impact on system performance

Table 17 Performance for various bias initializations, $\zeta = 40$

Carrier	Shipper		
	RS Bias	RN Bias	RA Bias
RS Bias	N/A [0.0]	0.18/0.69 [0.82]	0.72/0.08 [0.99]
RN Bias	0.02/0.91 [0.63]	0.39/0.54 [0.99]	0.88/−0.01 [0.98]
RA Bias	0.00/0.94 [0.96]	0.35/0.51 [0.92]	0.74/0.05 [0.89]

Each cell reports 'reward share carrier/reward share shipper [utilization]'. Agents use risk-averse learning rates and -penalty functions; only bias initializations deviate

Table 18 Performance for various bias initializations, $\zeta = 300$

Carrier	Shipper		
	RS	RN	RA
RS	−0.02/0.87 [0.13]	0.09/0.73 [0.92]	0.69/0.21 [0.99]
RN	−0.09/0.98 [0.78]	0.32/0.60 [0.98]	0.95/−0.08 [0.99]
RA	0.01/0.89 [0.98]	0.32/0.53 [0.92]	0.73/0.06 [0.67]

Each cell reports 'reward share carrier/reward share shipper [utilization]'. Agents use risk-averse learning rates and -penalty functions; only bias initializations deviate

Fig. 12 Carrier RN bias and shipper RN bias ($\zeta^C = 40$)

their initial bias (colloquially the opening bid/ask), which to a large degree determines the eventual convergence. Fairness is hard to achieve, but the best-case scenarios show almost full capacity utilization and a Nash adherence of about 85%.

7 Conclusions

This paper presents a policy gradient algorithm to explore strategic bidding behavior of shippers and carriers in self-organizing logistics. The research aligns with contemporary

Fig. 13 Carrier RN bias and shipper RS bias ($\zeta^C = 40$)

Fig. 14 Carrier RA bias and shipper RS bias ($\zeta^C = 300$)

freight platforms, while providing a building block of the Physical Internet. When moving towards automated negotiations and dynamic resource utilization, bidding strategies have a major impact on system stability. In our model-free learning interpretation, agents only observe whether bids/asks are accepted or rejected, learning and adapting their strategy based on this limited information while their opponent does the same. As strategies may be updated at any time, the learning problem is online and non-stationary.

To minimize behavioral assumptions, we propose a deep reinforcement learning algorithm rooted in policy gradient theory to learn strategies for bidding and asking. Inspired by financial markets, a neutral rule-based broker (which may be viewed as the environment by carrier and shipper) schedules jobs at a batch level by maximizing bid-ask spreads. This mechanism

ensures that the highest bidders and most economical transport services get preference, which would be desirable in real life.

We perform a number of numerical experiments, analyzing the results based on desired properties in bargaining games. Adherence to Nash equilibria, utilization of transport capacity and fairness are defined as key performance metrics. Any solution that divides the market value (difference between maximum willingness to pay and transport costs) between carrier and shipper is a non-cooperative Nash equilibrium, and as such may be seen as an optimal solution. Approximating the Nash equilibrium is only possible with high utilization. A fair division between of market value between agents is not requisite for an equilibrium, but is reflected in both normative game-theoretical solutions and practical settings. From an individual perspective, the agents attempt solely to maximize their own rewards.

In a deterministic test case, the ideal market situation is approximated quite closely. With both agents adopting comparable learning algorithms, we obtain stable markets that score well on the KPIs, with $\sim 99\%$ of jobs being shipped and Nash adherence and fairness metrics being well over 90% (the stochastic nature of the policies prevents perfect Nash equilibria). When varying the agents' risk profiles, we find that being more risk-seeking (in terms of opening bid and ask, penalizing failed negotiations and responsiveness to changes) than the opponent is generally rewarded. This gives agents a strong incentive to strategically place bids and asks. Fairness suffers as a result, although utilization and Nash adherence remain high.

In a stochastic case, the best attained Nash adherence is $\sim 85\%$. As the market is more uncertain, bid-ask spreads increase and the broker takes a larger reward share. We find that cautious strategy updates are necessary to preserve system stability. To improve individual gains, proper initialization of bids and asks is essential. Bold opening bids and asks tend to yield higher rewards, yet if both agents adopt the same strategy results are poor for all included, illustrating the strategic complexities of the environment.

For both test cases, the ability to improve one's reward share by risk-seeking behavior has a potential drawback from a system perspective. The results show that—especially if both agents engage in overly risky behavior—the system performance may strongly decline or even cease to be stable. Being unable to observe the opponent's strategy, this causes concern when designing fully decentralized transport markets. From a game-theoretical perspective, risk-seeking behavior might be avoided to safeguard long-term rewards.

Overall, the results are encouraging as a stepping stone towards decentralized and self-organizing freight transport markets. In particular, the work could be extended towards multiple shippers and carriers, such that a variety of strategies competes. After some calibration efforts, we consistently obtain solutions that approximate the Nash equilibria describing optimal markets. By appropriately designing their bidding- and asking algorithms, market actors can embed their real-life preferences and risk appetite. The neutral broker appropriately assigns jobs, taking a larger reward share in markets characterized by uncertainty. Generally, we expect to see such properties in a healthy market. The evaluated algorithmic design results in a well-functioning and self-organized market without reliance on contracts, regulations or communication protocols. For the real-world transport market platforms that gain prominence nowadays, this implies that a strong central party and dense regulation are not essential, potentially easing the design of such platforms. However, it also highlights that competitive unbalance may arise due to algorithmic advances. For long term visions such as the Physical Internet, the research demonstrates a rational basis for completely decentralized transport markets.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Appendix A: Proofs

Lemma 1 (Job selection when bid price is lower than ask price) *If $b_{n,j} < a_{n,j}$, the broker will always set $\gamma_{n,j} = 0$ to maximize its profits.*

Proof Take any pair $(b_{n,j}, a_{n,j}) \in \mathbb{R}^2$ satisfying $b_{n,j} < a_{n,j}$. Then $\gamma_{n,j}(b_{n,j} - a_{n,j}) < 0$ if $\gamma_{n,j} = 1$ and $\gamma_{n,j}(b_{n,j} - a_{n,j}) = 0$ if $\gamma_{n,j} = 0$. As Eq. (3) is a maximization function, it follows that $\gamma_{n,j} = 0, \forall b_{n,j} < a_{n,j}$. $\square$

Lemma 2 (Existence of disagreement point) *For the carrier, there exists a strategy π^C that ensures the shipper's payoff r_j^S equals at most 0 for any job j , regardless of the opponent's strategy π^S .*

Proof Take any job j . Let $a_{n,j} \in \mathbb{R}$ be the ask price. If $c_j^{C, trn} \leq a_{n,j} \leq b_{n,j}$ is satisfied, the carrier earns $a_{n,j} - c_j^{min} \geq 0$ if $\gamma_{n,j} = 1$ and 0 if $\gamma_{n,j} = 0$ [see Eq. (2)]. As both strategies map onto the real domain, there exists pairs of strategies (π^C, π^S) that ensure nonnegative payoffs for the carrier. The carrier needs not to invoke the disagreement point in this case.

Now suppose $a_{n,j} \leq b_{n,j} < c_j^{C, trn}$. In this case, the carrier earns $a_{n,j} - c_j^{C, trn} < 0$ if $\gamma_{n,j} = 1$ and can achieve at most 0 (when $\gamma_{n,j} = 0$). It follows that if $b_{n,j} < c_j^{C, trn}$, the carrier is better off asking $a_{n,j}$ such that $b_{n,j} < c_j^{C, trn} \leq a_{n,j}$, which by Lemma 1 always guarantees that $\gamma_{n,j} = 0$, yielding a payoff of 0 for both carrier and shipper. To execute the threat, it is not necessary that $b_{n,j} < c_j^{C, trn} \leq a_{n,j}$ always holds. Following Eq. (1), if π^C yield $a_{n,j} \geq c_j^{S, max}$, the shipper's reward r_j^S can be at most 0. If $b_{n,j} \geq a_{n,j} \geq c_j^{S, max}$, the shipper receives either $c_j^{S, max} - b_{n,j} \leq 0$ or 0. If $b_{n,j} < a_{n,j}$, the payoff is by default 0 (Lemma 1). Thus, setting $a_{n,j} \geq c_j^{S, max}$ guarantees that the payoff of the shipper is at most 0; a result holding for all bids $b_{n,j} \in \mathbb{R}$ and thus for all strategies $\pi^S \in \Pi^S$. $\square$

Lemma 3 (Nash equilibrium) *Any payoff profile satisfying $r_j^C + r_j^S \equiv c_j^{S, max} - c_j^{C, trn}$ is a Nash equilibrium. This payoff profile is achieved by any pair of strategies satisfying $c_j^{C, trn} \leq a_{n,j} = b_{n,j} \leq c_j^{S, max}$.*

Proof Take any job j . We first show that rational agents will adopt strategies that satisfy $c_j^{C, trn} \leq a_{n,j} \leq b_{n,j} \leq c_j^{S, max}$, being the only profile that may yield positive payoffs. After establishing that this is the feasible set, we continue to prove the Nash equilibrium.

It cannot be guaranteed that $\gamma_{n,j} = 1$ for any bid-ask pair; from Eq. (3) it follows this decision also depends on availability of sufficient capacity. However, $\gamma_{n,j} = 0$ yields a payoff of 0, which is equivalent to the disagreement point. For this reason, we always refer to payoffs ≥ 0 or ≤ 0 . We distinguish between four sets of payoff profiles.

Set 1 (not feasible) : $b_{n,j} < c_j^{C, trn}$

The carrier cannot obtain a positive payoff with any $a_{n,j}$ satisfying $a_{n,j} \leq b_{n,j} < c_j^{C, trn}$. From Lemma 2, it is therefore beneficial for the carrier to execute the threat, ensuring a payoff of 0 for both agents.

Set 2 (not feasible) : $a_{n,j} > c_j^{S, max}$

The shipper cannot obtain a positive payoff with any $b_{n,j}$ satisfying $b_{n,j} \geq a_{n,j} > c_j^{S, max}$. From Lemma 1, it is beneficial for the shipper to execute the threat, ensuring a payoff of 0 for both agents.

Set 3 (not feasible) : $c_j^{C, trn} \leq b_{n,j} < a_{n,j} \leq c_j^{S, max}$

From Lemma 1, any bid-ask pair for which $b_{n,j} < a_{n,j}$ ensures a payoff of 0 for both agents.

Set 4 (feasible) : $c_j^{C, trn} \leq a_{n,j} \leq b_{n,j} \leq c_j^{S, max}$

The carrier may earn $a'_{n,j} - c_j^{C, trn} \geq 0$ and the shipper may earn $c_j^{S, max} - b'_{n,j} \geq 0$. Both agents have no incentive to invoke the disagreement point as payoffs are greater than or equal to 0.

We established that Set 4 is the only feasible set; positive payoffs can only be attained when adhering to $c_j^{trn} \leq a_{n,j} \leq b_{n,j} \leq c_j^{max}$. We now show that $r_j^C + r_j^S = c_j^{S, max} - c_j^{C, trn}$ must be satisfied for a Nash equilibrium. For a proof by contradiction, suppose $c_j^{C, trn} \leq a_{n,j} < b_{n,j} \leq c_j^{S, max}$ holds. The carrier may earn $a_{n,j} - c_j^{C, trn} \geq 0$. However, it could unilaterally improve its payoff by selecting any $a'_{n,j}$ satisfying $a_{n,j} < a'_{n,j} \leq b_{n,j}$, yielding a potential payoff $a'_{n,j} - c_j^{C, trn} > a_{n,j} - c_j^{C, trn}$ without triggering the disagreement point. Similarly, the shipper may select any $b'_{n,j}$ satisfying $a_{n,j} \leq b'_{n,j} < b_{n,j}$, potentially yielding $c_j^{S, max} - b'_{n,j} > c_j^{S, max} - b_{n,j}$ without triggering the disagreement point. It follows that $c_j^{C, trn} \leq a_{n,j} = b_{n,j} \leq c_j^{S, max}$ is the equilibrium point. Any unilateral deviation would either reduce the agent's own payoff or invoke the disagreement point. Under the condition that $a_{n,j} = b_{n,j}$, subject to $c_j^{C, trn} \leq a_{n,j}$ and $b_{n,j} \leq c_j^{S, max}$, the payoff is $(a_{n,j} - c_j^{C, trn}) + (c_j^{S, max} - b_{n,j}) = (a_{n,j} - c_j^{C, trn}) + (c_j^{S, max} - a_{n,j}) = c_j^{S, max} - c_j^{C, trn}$; exactly the full system gain is divided between the agents. This result is equivalent to the equilibrium for the bargaining game as defined by Nash (1953). $\square$

Appendix B: Experimental results for actor network architectures

This appendix shows the results of various actor network architectures, corresponding to Sect. 6.1.5. See Table 19 for the results per network architecture.

Appendix C: Bias initialization

At the start of each experiment, we initialize the bias weight, with the bias node itself being a node with value 1. With all other weights at the output layer being initialized at 0, this means that the initial bids and asks are equal to the bias weights provided. Informally the bias initialization can be seen as the opening bid or ask, greatly influencing the eventual convergence.

The costs and willingness to pay for each job depend on two job characteristics: distance and volume. Let $\mathbf{d} = [1, \dots, d^{max}]$ be the vector of feasible distances and $\mathbf{v} = [1, \dots, \zeta^{max}]$

Table 19 Comparison of various actor network architectures

Architecture	Utilization	Nash adherence	Fairness
Linear	0.99 (1.00)	0.96 (0.97)	0.86 (0.97)
1L5N	0.99 (1.00)	0.93 (0.95)	0.64 (0.85)
1L10N	0.99 (0.99)	0.92 (0.92)	0.93 (0.93)
1L15N	0.99 (0.99)	0.91 (0.92)	0.85 (0.94)
1L20N	0.99 (1.00)	0.91 (0.91)	0.95 (0.95)
1L25N	0.99 (0.99)	0.91 (0.89)	0.92 (0.96)
1L30N	0.99 (0.99)	0.90 (0.92)	0.94 (0.96)
2L5N	0.99 (0.99)	0.90 (0.93)	0.60 (0.93)
2L10N	0.99 (0.99)	0.88 (0.91)	0.91 (0.94)
2L15N	0.99 (0.99)	0.89 (0.90)	0.93 (0.94)
2L20N	0.99 (0.99)	0.89 (0.91)	0.92 (0.94)
2L25N	0.99 (0.99)	0.89 (0.93)	0.91 (0.90)
2L30N	0.87 (0.99)	0.76 (0.91)	0.60 (0.89)
3L5N	0.99 (0.99)	0.91 (0.92)	0.88 (0.86)
3L10N	0.99 (0.99)	0.89 (0.94)	0.94 (0.94)
3L15N	0.99 (0.99)	0.89 (0.94)	0.94 (0.94)
3L20N	0.99 (0.99)	0.89 (0.93)	0.92 (0.96)
3L25N	0.74 (1.00)	0.61 (0.92)	0.22 (0.35)
3L30N	0.94 (0.99)	0.85 (0.94)	0.82 (0.88)

In general, we observe worse average results for deep networks and worse average results for large numbers of nodes. End-of-horizon values are generally better than for smaller networks

be the vector of feasible volumes. By taking the outer product of the vectors, we obtain a matrix containing the marginal weight of each job type:

$$\mathbf{d} \otimes \mathbf{v} = \begin{bmatrix} d_1 v_1 & d_1 v_2 & \dots & d_1 v_{\zeta^{max}} \\ d_2 v_1 & d_2 v_2 & \dots & d_2 v_{\zeta^{max}} \\ \vdots & \vdots & \ddots & \vdots \\ d_{d^{max}} v_1 & d_{d^{max}} v_2 & \dots & d_{d^{max}} v_{\zeta^{max}} \end{bmatrix}.$$

Recall that $c^{C, trn}$ describes the transport cost per volume unit per distance unit and $c^{S, max}$ the willingness to pay. The transport cost matrix—containing transport costs for each job type—is then given by $(\mathbf{d} \otimes \mathbf{v}) \cdot c^C$. Similarly, we may retrieve the willingness to pay for each job type by computing $(\mathbf{d} \otimes \mathbf{v}) \cdot c^{S, max}$. As we assume independent uniform probability distributions for every job characteristic, each matrix element has an equal probability. Thus,

Table 20 Bias initializations per risk profile

Risk profile	Carrier	Shipper
Risk-seeking	$\bar{c}^{S, max}$	$\bar{c}^{C, trn}$
Risk-neutral	$(\bar{c}^{C, trn} + \bar{c}^{S, max})/2$	$(\bar{c}^{C, trn} + \bar{c}^{S, max})/2$
Risk-averse	$\bar{c}^{C, trn}$	$\bar{c}^{S, max}$

Table 21 Normalized net rewards per agent

Carrier	Shipper		
	RS	RN	RA
RS	0.00/0.00	<u>0.69/0.15</u>	<u>0.78/0.13</u>
RN	<u>0.24/0.73</u>	0.46/0.49	0.61/0.28
RA	<u>0.80/0.83</u>	0.26/0.62	0.41/0.40

Best individual responses (Case I) to an opponent's strategy are underlined, Nash equilibria are in bold

Table 22 Normalized net rewards per agent

Carrier	Shipper		
	RS Bias	RN Bias	RA Bias
RS Bias	0.00/0.00	0.15/0.57	0.08/ <u>0.67</u>
RN Bias	<u>0.01/0.57</u>	<u>0.39/0.53</u>	<u>0.87/−0.01</u>
RA Bias	<u>0.00/0.90</u>	0.32/0.47	0.66/0.05

Best individual responses (Case II, $\zeta = 40$) to an opponent's strategy are underlined, Nash equilibria are in bold

Table 23 Normalized net rewards per agent

Carrier	Shipper		
	RS Bias	RN Bias	RA Bias
RS Bias	0.00/0.11	0.08/ <u>0.67</u>	0.68/0.21
RN Bias	<u>−0.07/0.76</u>	<u>0.31/0.59</u>	<u>0.94/−0.08</u>
RA Bias	<u>0.01/0.87</u>	0.29/0.49	0.49/0.04

Best individual responses (Case II, $\zeta = 300$) to an opponent's strategy are underlined, Nash equilibria are in bold

the average transport costs and average willingness to pay are simple the averages of all elements in the respective matrices.

We use the average transport cost $\bar{c}^{C, trn}$ and average willingness to pay $\bar{c}^{S, max}$ to initialize the bias weights. To provide some intuition: if the shipper would set $\bar{c}^{C, trn}$ as the initial bias weight, it would on average pose bids equal to the average transport costs for all jobs. As we use a single actor network for all jobs, the bias is a constant that holds for all job types. At the risk of being superfluous, we stress that the network must still learn the values corresponding to the various job types.

The initial bias weights are formalized in Table 20.

Appendix D: Nash equilibria

This appendix investigates the existence of pure Nash equilibria when agents adopt asymmetric strategies. Assuming the opponent's strategy is fixed, an agent can determine its best response to that strategy. When both agents cannot improve rewards by unilaterally changing their strategy, an equilibrium is reached. There may be multiple equilibria. The results—for Case I, Case II ($\zeta = 40$) and Case II ($\zeta = 300$) respectively—are found in Tables 21, 22 and 23. The tables contain the normalized net rewards per agent (which depend on both the reward shares and the number of jobs shipped) and the best responses per agent.

Appendix F: Features

The input for the actor- and critic networks is a vector $\phi(j, J_n)$. For each job j , a unique input vector is computed to determine its bid- or ask price.

For all experiments, the features as tested and proposed in Van Heeswijk (2019) are used, expanded with the ask/bid price for the critic network. The features contained in vector ϕ are detailed in Table 24.

Table 24 Features describing the key state attributes, used as input for the actor- and critic networks

Feature	Description
1	Bias
j_τ	Job due date
j_d	Job transport distance
j_v	Job volume
$\frac{\sum_{j \in J_n} j_\tau}{ J_n }$	Average due date
$\frac{\sum_{j \in J_n} j_d}{ J_n }$	Average distance
$\frac{\sum_{j \in J_n} j_v}{ J_n }$	Average volume
$\sum_{j \in J_n} j_v$	Total volume
$ J_n $	Total number of jobs
$a_{n,j}^m$	Ask (only for Q-value carrier)
$b_{n,j}^m$	Bid (only for Q-value shipper)

References

Ambra, T., Caris, A., & Macharis, C. (2019). Towards freight transport system unification: Reviewing and combining the advancements in the physical internet and synchromodal transport research. *International Journal of Production Research*, 57(6), 1606–1623.

Atasoy, B., Schulte, F., & Steenkamp, A. (2020). Platform-based collaborative routing using dynamic prices as incentives. *Transportation Research Record*, 2674(10), 670–679.

Aumann, R. J., & Shapley, L. S. (1994). Long-term competition-a game-theoretic analysis. In N. Megiddo (Ed.), *Essays in game theory* (pp. 1–15). Springer.

Cruijssen, F. (2020). Cross-chain collaboration in logistics. In *International series in operations research and management science (ISOR)* vol. 297.

Friedman, J. W. (1971). A non-cooperative equilibrium for supergames. *The Review of Economic Studies*, 38(1), 1–12.

Grondman, I., Busoniu, L., Lopes, G. A., & Babuska, R. (2012). A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6), 1291–1307.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (pp. 1026–1034).

Kalai, E. (1977). Proportional solutions to bargaining situations: Interpersonal utility comparisons. *Econometrica: Journal of the Econometric Society*, 45(7), 1623–1630.

Kalai, E., & Smorodinsky, M. (1975). Other solutions to Nash’s bargaining problem. *Econometrica: Journal of the Econometric Society*, 43(3), 513–518.

Karam, A., Reinau, K. H., & Østergaard, C. R. (2021). Horizontal collaboration in the freight transport sector: Barrier and decision-making frameworks. *European Transport Research Review*, 13(1), 1–22.

- Kellerer, H., Pferschy, U., & Pisinger, D. (Eds.). (2004). Multidimensional knapsack problems. In *Knapsack problems* (pp. 235–283). Springer.
- Kenney, M., & Zysman, J. (2016). The rise of the platform economy. *Issues in Science and Technology*, 32(3), 61.
- Klapp, M. A., Erera, A. L., & Toriello, A. (2018). The one-dimensional dynamic dispatch waves problem. *Transportation Science*, 52(2), 402–415.
- Miller, J., & Nie, Y. M. (2020). Dynamic trucking equilibrium through a freight exchange. *Transportation Research Part C: Emerging Technologies*, 113, 193–212.
- Minkoff, A. S. (1993). A Markov decision model and decomposition heuristic for dynamic vehicle dispatching. *Operations Research*, 41(1), 77–90.
- Montreuil, B. (2011). Toward a Physical Internet: Meeting the global logistics sustainability grand challenge. *Logistics Research*, 3(2–3), 71–87.
- Montreuil, B., Meller, R. D., & Ballot, E. (2013). *Physical internet foundations* (pp. 151–166). Berlin Heidelberg.: Springer.
- Nash, J. (1950). The bargaining problem. *Econometrica*, 8, 155–162.
- Nash, J. (1953). Two-person cooperative games. *Econometrica: Journal of the Econometric Society*, 21(1), 128–140.
- Qiao, B., Pan, S., & Ballot, E. (2019). Dynamic pricing model for less-than-truckload carriers in the Physical Internet. *Journal of Intelligent Manufacturing*, 30(7), 2631–2643.
- Rolnick, D., & Tegmark, M. (2017). The power of deeper networks for expressing natural functions. arXiv preprint [arXiv:1705.05502](https://arxiv.org/abs/1705.05502).
- Rubinstein, A. (1980). Strong perfect equilibrium in supergames. *International Journal of Game Theory*, 9(1), 1–12.
- Rubinstein, A. (1994). Equilibrium in supergames. In N. Megiddo (Ed.), *Essays in game theory* (pp. 17–27). Springer.
- Sallez, Y., Pan, S., Montreuil, B., Berger, T., & Ballot, E. (2016). On the activeness of intelligent physical internet containers. *Computers in Industry*, 81, 96–104.
- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). Cambridge, Massachusetts: The MIT Press.
- Van Heeswijk, W. J. A. (2019). Smart containers with bidding capacity: A policy gradient algorithm for semi-cooperative learning. In E. Lalla-Ruiz, M. R. K. Mes, & S. Voß (Eds.), *Lecture notes in computer science* (Vol. 12433, pp. 469–491). Cham: Springer.
- Van Heeswijk, W. J. A., & La Poutre, H. (2018). Scalability and performance of decentralized planning in flexible transport networks. In *2018 IEEE International Conference on Systems Man, and Cybernetics* (pp. 292–297). Piscataway, New Jersey: Institute of Electrical and Electronics Engineers Inc.
- Van Heeswijk, W. J. A., Mes, M. R. K., & Schutten, J. M. J. (2019a). The delivery dispatching problem with time windows for urban consolidation centers. *Transportation Science*, 53(1), 203–221.
- Van Heeswijk, W. J. A., Mes, M. R. K., & Schutten, J. M. J. (2019). Transportation management. In H. Zijm, M. Klumpp, A. Regattieri, & S. Heragu (Eds.), *Operations, Logistics and supply chain management* (pp. 469–491). Cham: Springer.
- Voccia, S. A., Campbell, A. M., & Thomas, B. W. (2019). The same-day delivery problem for online purchases. *Transportation Science*, 53(1), 167–184.
- Wang, Y., Nascimento, J. M. D., & Powell, W. B. (2018). Reinforcement learning for dynamic bidding in truckload markets: an application to large-scale fleet management with advance commitments. arXiv preprint [arXiv:1802.08976](https://arxiv.org/abs/1802.08976).
- Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4), 229–256.
- Yan, F., Ma, Y., Xu, M., & Ge, X. (2018). Transportation service procurement bid construction problem from less than truckload perspective. *Mathematical Problems in Engineering*, 2018, 1–17.
- Zha, L., Yin, Y., & Du, Y. (2017). Surge pricing and labor supply in the ride-sourcing market. *Transportation Research Procedia*, 23, 2–21.