Talking Trucks: Decentralized Collaborative Multi-Agent Order Scheduling for Self-Organizing Logistics

Geert L.J. Pingen[1], Christian R. van Ommeren[1], Cornelis J. van Leeuwen[1], Ruben W. Fransen[1], Tijmen Elfrink[1], Yorick C. de Vries[1,2], Janarthanan Karunakaran[3], Emir Demirović[2], Neil Yorke-Smith[2]

[1]The Netherlands Organisation for Applied Scientific Research (TNO), The Hague, The Netherlands
[2]Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, The Netherlands
[3]Van Berkel Logistics B.V., Veghel, The Netherlands
{geert.pingen, christian.vanommeren, coen.vanleeuwen, ruben.fransen, tijmen.elfrink}@tno.nl,
y.c.devries@tudelft.nl, jkarunakaran@vanberkellogistics.eu, {e.demirovic, n.yorke-smith}@tudelft.nl

Abstract

Logistics planning is a complex optimization problem involving multiple decision makers. Automated scheduling systems offer support to human planners; however state-of-the-art approaches often employ a centralized control paradigm. While these approaches have shown great value, their application is hindered in dynamic settings with no central authority. Motivated by real-world scenarios, we present a decentralized approach to collaborative multi-agent scheduling by casting the problem as a Distributed Constraint Optimization Problem (DCOP). Our model-based heuristic approach uses message passing with a novel pruning technique to allow agents to cooperate on mutual agreement, leading to a near-optimal solution while offering low computational costs and flexibility in case of disruptions. Performance is evaluated in three real-world field trials with a logistics carrier and compared against a centralized model-free Deep Q-Network (DQN)-based Reinforcement Learning (RL) approach, a Mixed-Integer Linear Programming (MILP)-based solver, and both human and heuristic baselines. The results demonstrate that it is feasible to have virtual agents make autonomous decisions using our DCOP method, leading to an efficient distributed solution. To facilitate further research in Self-Organizing Logistics (SOL), we provide a novel real-life dataset.

Introduction

Logistics planning is arguably the most complex task in supply chain management. In current-day logistics, human planners function as orchestral conductors, orchestrating assets and directing people to complete deliveries on time. It is therefore not surprising that algorithmic approaches classically perform centralized optimization and scheduling (ter Mors, Zutt, and Witteveen 2007).

Such centralization is problematic, however, when multiple parties are involved in the supply chain: it assumes data can be sufficiently centralized, and precludes localized autonomy, for example when local disruptions occur. Moreover, large real-world scheduling problems can become intractable for centralized solvers and place a heavy demand on compute resources.

Self-organizing logistics (SOL) systems provide an alternate approach. A SOL system is an agent-based system in which agents can be actors (e.g., shippers) or assets (e.g., trucks). Although there is a common goal, a SOL system should respect the individual constraints and objectives of its agents (Pan, Trentesaux, and Sallez 2016).

This paper studies the real-life logistics problem of *order scheduling*, which resembles a combination of job-shop scheduling and vehicle routing with time-window constraints. Building on previous work of van Ommeren et al. (2020), we provide a decentralized model-based method of solving the logistics problem with three centralized alternatives: human domain expert planning; model-free RL planning; and model-based planning executed by a MILP-based solver. In addition we incorporate two decentralized heuristic-based baselines. Cooperative Constraint Approximation (CoCoA) (van Leeuwen and Pawełczak 2017) is applied in a novel logistics setting by adding a new pruning step to regulate complexity. These methods were examined in three field-trials with real order scheduling problems.

Embedding these methods in an operational context brings great challenges, because logistics is characterised by supply chain dynamics and an environment of continuous change. In real-life, human planners monitor and evaluate many variables and flexible objective functions that go far beyond straightforward cost optimization.

We show that it is feasible in real-world scenarios to yield autonomy to digital twins representing actual vehicles, and to enable autonomous decision-making on order scheduling in a decentralized collaborative fashion. In doing so, the agents can generate more efficient solutions than human planners while incorporating driver preferences, truck characteristics, and many additional constraints, such as interdependence between rides.

The contributions of this paper to the literature are:

1. A novel decentralized collaborative agent-based method to the order scheduling problem that outperforms human baselines on punctuality Key Performance Indicators (KPIs) – validated in real-world experiments.
2. The incorporation of feedback from human planners on generated allocations as an integral part of the algorithm development cycle and the decision support system.

3. A comparison with a centralized model-free DQN-based RL approach, a MILP-based solver, and heuristic baselines.
4. A novel real-world scheduling dataset for benchmarking SOL solutions, publicly available at: https://zenodo.org/record/5777315.

Related Work

Order Scheduling Problem

The *order scheduling* problem is a combination of a classical Vehicle Routing Problem (VRP) (Dantzig and Ramser 1959) and a Job-Shop Problem (JSP). Orders have a list of stops. The handling time at each stop and driving time between the stops together make the processing time of a 'job', which can be processed by a vehicle. The first stop and last stop of an order can be different locations, therefore the problem can also be considered a VRP. To optimize globally from the trucking company's point of view, the problem with time windows can be formulated as a linear program, when using constraint relaxation. Subsequently it can be solved with various exact algorithms (Baldacci, Mingozzi, and Roberti 2012). Meta-heuristic methods are widely used, e.g.: neighbourhood search (Song et al. 2020). Chao (2002) presents a formulation for a truck and trailer assignment problem: a VRP with similar truck and trailer assignment decisions is presented and approached with tabu search.

Using global problem formulations to solve the order scheduling problem has some notable disadvantages. Centralized planning is difficult to scale to larger order or truck sets. Additionally, centralized formulations prefer homogeneous constraints and objectives for orders and trucks in the assignment problem. In practice, such homogeneity rarely exists. Extensive coordination between stakeholders is required to align on common interfaces.

A different way of approaching the order scheduling problem is by devolving the level of decision making, from a central point at a trucking company, to making the assignment decision at, e.g., the truck level. This leads to a *decentralized* decision structure known as SOL (Gerrits 2020). Quak, van Kempen, and Hopman (2018) claim that SOL can be considered an intermediate solution between central organization and full decentral organization.

We are not the first to implement decentralized decision-making in logistics (Berndt 2011; Feng et al. 2017; Gerrits 2020). To our knowledge, however, we are the first to apply and compare the decentralized method on a real-life case and show the value of decentralized decision making compared to current practice and centralized approaches. Further, to our knowledge we present the first implementation of the CoCoA negotiation algorithm in a logistics application.

Distributed Constraint Optimization

DCOPs are a class of distributed optimization problems where discrete variables have to be assigned values, in such a way that given constraints between the variables are satisfied (Hirayama and Yokoo 1997). Each variable is controlled by a separate agent. Constraint violations incur cost (in $\mathbb{R}_{\geq 0}$). Agents co-operate by passing messages back and forth to find assignments that minimize the sum of all constraint costs.

Since DCOPs are $\mathbf{NP}$-hard, finding the optimal solution is often not feasible for more than a few variables. Improving scalability of DCOPs is an active research topic (Yeoh 2018). Thus we use a local search method, the *CoCoA algorithm* (van Leeuwen and Pawełczak 2017). The advantages of CoCoA are that it is a non-iterative single step look-ahead algorithm, and that it is capable of dealing with asymmetric constraints, where the costs of an assignment differ among the agents involved (Grinshpoun et al. 2013). Using this strategy, it is possible to find assignments that are practically feasible.

Reinforcement Learning

Model-free RL algorithms such as DQNs (Mnih et al. 2013) or Proximal Policy Optimization (Schulman et al. 2017) that do not require explicit modelling of transition functions and reward signals, are a promising alternative to methods where utility functions or rules need to be manually constructed by experts. A main problem in model-free RL approaches, however, is their dependence on representative training data, sample efficiency, and high computational cost (Nguyen, Nguyen, and Nahavandi 2020).

Gombolay et al. (2018) highlight the value of incorporating human expertise and heuristics in RL performance on VRPs. Joe and Lau (2020) show that combining RL and a meta-heuristic is effective for centralized solving of dynamic VRP problems. In the SOL domain, Irannezhad, Prato, and Hickman (2020) successfully incorporate a multi-agent RL solution into a full-fledged port decision support system and evaluate agent collaboration strategies, showing that a cooperative strategy, rather than one focused on individual reward maximization, results in the highest overall vehicle utilization and lowest travel distance and costs. Tang et al. (2021) improve on another model-free RL method, Soft Actor-Critic (SAC) (Haarnoja et al. 2018) – an off-policy model like DQN – with improved sample efficiency by incorporating an entropy factor and regularization. The authors apply it to an unmanned warehouse environment where autonomous robotic platforms are used for order-picking.

Problem and Solution Methods

Problem Formulation

The order scheduling problem is a combination of a VRP and JSP. The problem is defined by sets of trucks V, trailer conditions C, orders O, locations L, driving times D, and a notion of time Ω. The general objective is to maximize the number of on-time deliveries of orders and minimize the number of used trucks. The problem is formulated thus:

Parameters:

- *Trucks* V: A truck v has driver working hours defined by time of start $t^S \in \Omega$ and time of end $t^E \in \Omega$, $W_t = [t^S, t^E]$. Further, start location $l^S \in L$, end location $l^E \in L$, truck-trailer state at start $c^S \in C$. We define a truck $v \in V$ as $v := (l^S, l^E, c^S, W_t)$

Method	Objectives (explicit)	Constraints	Design decisions
HUMAN	Maximize number of orders delivered, order punctuality, and adherence to driver preferences; minimize number of used trucks.	**Drivers**: Working hours, preferences, certifications. **Trucks**: Type (port, region, terminal), emission category, sleeper cabin (no overnight stays), chassis. **Orders**: Service type (decoupling, live-handling). **Customers**: Preferences w.r.t. strictness of delivery time.	Learned driving and handling time (from experience); Order allocation process is similar to earliest deadline first.
RAND & EAR-D	Maximize number of orders delivered.	**Trucks** same as HUMAN. For **Drivers**, only starting time is considered. **Orders** not before start of delivery window.	Randomized assignment for RAND, greedy selection based on earliest deadline for EAR-D. Great circle distance based driving time.
SOLV	Maximize number of orders delivered.	Same as HUMAN, except for **Orders** where it uses no decoupling service type, and fixed deadlines rather than delivery windows and no **Customer**-specific constraints.	Late orders are not allowed. Incorporates mandatory breaks and empty return trips after delivery. Driving time based on Google Maps, historical data, and a slack factor.
AGT-DCOP	Maximize number of orders delivered; minimize number of used trucks.	Same as HUMAN, except strict **Order** delivery windows and no **Customer**-specific constraints.	Late orders can not occur. Great circle distance based driving time.
RL	Maximize order punctuality.	**Trucks** same as HUMAN. For **Drivers**, only starting time is considered.	Early orders are not allowed but late orders are. Great circle distance based driving time.

Table 1: Differences in objectives, constraint modelling, assumptions and design decisions for all methods.

- *Orders O*: In practice, an order is a series of locations and a delivery time window at each location. We define an order by the first location $l^S \in L$, arrival time window at first location W^S, start condition $c^S \in C$, last location $l^E \in L$, end condition $c^E \in C$ and order handling duration δ from first to last location: $o := (l^S, l^E, c^S, c^E, W^S, \delta)$ In practice orders can consist of multiple location visits with individual time windows, we aggregate this to a start and end location, and a single time window at the start location W^S.
- *Trailer conditions C*: a truck can have no trailer, an empty trailer or a trailer with a container.
- *Driving times*: $d_{i,j} \in D$ between locations $l_i, l_j \in L$.

Decision Variables:
The problem is to assign a sequence of orders $o_i \in O$ to each truck $v \in V$. The arrival time λ_i at each order o_i depends on the sequence of the orders, the start time, driving time to each subsequent order and the driving time to the truck's end location. The sequence of assigned orders to a truck v is noted as A_v and defines driving times $\hat{d}_v^A$ with $A_v := ((o_i, \lambda_i), (o_j, \lambda_j), ..., (o_n, \lambda_n))$ and $\hat{d}_v^A := (d_{S,i}, d_{i,j}, ..., d_{n,E})$. Note that the trailer state of truck c can change because of different order types, (de)coupling or live (un)loading at customer. Picking-up or dropping an empty trailer at the terminal is added as an optional order to O which allows trucks to change trailer state, incurring driving time to home terminal and (de)coupling time.

Objective: The objective is to maximize the number of on-time delivered orders. $max \sum_{v \in V} |A_v|$. Empty trailer orders do not count for this objective.

Constraints: For all trucks, $\forall v \in V$:

- At most one order per truck at each point in time. $\lambda_{i+1} = \lambda_i + \delta_i + d_{i,i+1} \; \forall o_i \in A_v$
- Each order's first stop must be made within its time window. $\lambda_i \in W_i^S \;\; \forall o_i \in A_v$
- All orders must be executed within truck working hours. For 1 being the first and n being last order in A_v: $\lambda_1 - d_{S,1} \in W_v$
  $\lambda_n + \delta_{o_n} + d_{n,E} \in W_v$
- The end trailer state and start trailer state of subsequent orders must match. $c_{i+1}^S = c_i^E$

This general formulation covers the most essential constraints of the problem. There are additional constraints on container size, container weight and location limitations for trucks. This means that some orders cannot be assigned to specific trucks. Further, note that the definition of the utility or reward evaluation is crucial. It might be possible to move constraints to the utility function to change the solution space.

The diverse methods used in this work apply slightly varying definitions. We explicitly outline these differences in Table 1 to make a balanced comparison. We harmonized constraints between all six methods within the operational character of this work. For human planners, constraints and objectives are naturally flexible. For our AGT-DCOP method, the goal was to incorporate as many real-world (HUMAN) constraints and actions as possible, while maintaining the capability to generate plannings in real-time. Any additional constraints not incorporated in the system are delegated to human decision makers. The RL method is implemented as a machine-learning alternative, and required minor relaxing of constraints to produce feasible plannings. The MILP-based solver is inherently different in design, and in active use by the logistics carrier. These were important reasons to include it in this work, despite it allowing limited control over model parameters.

Baseline Approaches

To quantitatively evaluate the decentralized agent-based DCOP and centralized model-free RL approaches, the following baselines are included.

Human Approach (HUMAN) A planner has two concrete tasks: validate that sufficient trucks are available the next day, and communicate orders to drivers for the current day.

The goal of both tasks is to ensure that orders reach their destinations at the time the customer desires. To do so, a planner follows a four-step process:

1. All orders are sorted based on their maximum departure time at the terminal and iteratively assigned to drivers by adding the order identifier and its desired delivery time to a spreadsheet. During this process the planner accounts for constraints (e.g. driver working hours) and driver preferences (e.g. long drives versus shuttling orders to the same customer).
2. Mornings are typically characterised by peak volume and insufficient truck capacity. A planner knows which customers allow deviation from the delivery time and uses this knowledge to ensure that time-critical orders are delivered on time.
3. At the start of the day, all first orders are communicated to the drivers and usually executed as planned.
4. Follow-up orders are assigned at the moment a truck is within a 30-minute radius of the inland-terminal when returning from a delivery. The initial planning receives only limited attention from the human planners: what matters most is ensuring containers arrive at their destinations in time.

Heuristic Approach (RAND & EAR-D) Two simple heuristic-based methods are included in the set of baselines. The first method is a straightforward random selection on the available orders whenever a truck is ready to pick up a container. This method is not expected to produce competitive results, but serves as a worst-case scenario benchmark.

The second, more intelligent heuristic, selects orders on the basis of earliest deadline. Whenever a truck is available, it will select the order that has the closest upcoming deadline for transport. Conflicts are mitigated by forcing agents to decide on orders sequentially through random ordering.

Centralized MILP Approach (SOLV) The final baseline is a centralized approach in active use at the logistics carrier. It is adapted from the MILP case in Karunakaran (2020) (Matching truck model) that uses a one-to-many matching assignment technique – greedily assigning as many orders as possible to a truck within the given delivery time windows to achieve maximum utilisation of the fleet. It incorporates mandatory 30 minute breaks for drivers after every 6 hours, driver preferences and license restrictions. A notable difference with other methods is that orders are forced to be a round-trip: trucks start from the origin (base terminal), serves a client, and returns back to the origin. This results in more conservative plannings. The objective function is given in Eq. (1). Full design decision details are listed in Table 1 and available in Karunakaran (2020).

$$max \frac{w_{max}}{w_t} \sum_{m=1}^{ub_{trips}} \sum_{o \in O} Y_{m,o} + \sum_{m=1}^{ub_{trips}} \sum_{o \in O} \theta_{m,o} \qquad (1)$$

where:

w_t	$=$ Utility cost of truck $t \in T$
w_{max}	$=$ Max. value of the utility cost of $trucks \in T$
ub_{trips}	$=$ Max. no. of daily trips possible for any truck
$Y_{m,o}$	$=$ Binary for delivery of order o in trip m
$\theta_{m,o}$	$=$ Binary for on-time delivery of order o in trip m
O	$=$ The set of orders

Decentralized Agent-Based Approach (AGT-DCOP)

Scheduling decisions at a truck level are implemented as agent logic. Agents have three main functionalities: communication; a scheduling heuristic to make daily schedules of orders; and preference negotiation. Communication is used to share information on available and assigned orders.

Best-First Search Daily Schedule Generation The scheduling heuristic is based on best-first search. In the search tree, each node is a daily schedule and new branches are added by iteratively adding orders to previously made daily schedules. For each node, a utility value is calculated with the formula below. To bound the search there is a maximum of 100 nodes at the start of each iteration, so only the most promising daily schedules are extended with new orders. This is considered to be a valid pruning method as daily schedules with a low utility, high waiting times or many kilometres without load, should be avoided.

The utility function was defined such that the best-first search leads schedules that maximize the number of scheduled orders. Utility (U) is based on valuing transport activity over the time it takes for execution, together with reducing deadline slack to valuate orders with tight time windows higher than orders with broader time windows:

$$U_A = \frac{\mu_A}{\lambda_A} + \frac{\mu_A}{\lambda_t^w} + w^o \cdot N_A^o + w^d \cdot f(\hat{\lambda}_A^d) - w^e \cdot \lambda_A^e \quad (2)$$

where:

u_o	$=$ Distance for order o
s_o	$=$ TEU(Twenty-foot Equivalent Unit) for order o
μ_A	$=$ Transport activity: $\Sigma_{o \in A} s_o \cdot u_o$
λ_A	$=$ Total duration of daily schedule A
λ_t^w	$=$ Total time of truck working day W^v
N_A^o	$=$ Number of orders in schedule A
$\hat{\lambda}_A^d$	$=$ Array of time slack before deadline per stop
$f(\hat{\lambda}_A^d)$	$=$ Function to prioritize small time windows
λ_A^e	$=$ Time driving without load
w^o	$=$ Weight factor for number of bookings
w^d	$=$ Weight factor for deadline utility
w^e	$=$ Weight factor for empty driving time

Cooperative Constraint Approximation (CoCoA) CoCoA works by having agents share the incurred costs of other agents' assignments, and then making a decision that is the best for themselves and their immediate neighbours. This heuristic strategy balances the solution costs and optimality in practice.

In order to represent the problem as a DCOP, the agents' top n schedules (s) of all trucks are used to construct a bipartite graph in which every truck is connected to every order (o) it considers taking. The order nodes are then collapsed and a graph is created, indicating which trucks have common interests in an order, and hence which trucks share a constraint. This *pruning* stage is critical, since simply connecting all trucks to all others would make the problem graph fully-connected, and hence infeasible to solve using a DCOP formulation.

Firstly, constraint costs for picking a different assignments can now be computed as follows:

$$\mathcal{C}_{i,j} = \begin{cases} \tau & \text{if } \exists o \; (o \in s_i) \cup (o \in s_j), \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where:

$\mathcal{C}_{i,j}$ = The cost between schedules for trucks i and j
τ = A large penalty factor for taking the same order

Ideally $\tau = \infty$, which would mean that not having equal assignments is a hard constraint; however, in CoCoA we cannot handle hard constraints, therefore we chose a large value such that $U^{\max} \ll \tau$, where $U^{\max}$ is the maximum utility.

Secondly, a preference constraint is added for every truck in which the utility of a schedule is expressed as the sum of the utilities of its orders according to Eq. (2).

A *no-assignment* utility of $10 \cdot U^{\max}$ is also added to the domain, allowing the option of not assigning a schedule at all. This indicates that no assignment is still better than a conflicting assignment. All truck agents are now connected to others through their shared constraints, the CoCoA algorithm can run, which will make them negotiate who gets which schedule.

A fully detailed description of the CoCoA algorithm is provided by (van Leeuwen and Pawełczak 2017), but in essence it comes down to the following. When an agent A is triggered, it sends messages to its neighbours (agents they share a constraint with) to gather information on the implied cost for the global utility when A picks any specific assignment. A neighbour B then replies with a message containing a *cost map*, which contains for any assignment for A, what the lowest cost for B is, given that assignment. Note, that what that assignment is, is not part of the reply, which increases the privacy for agent B. When all cost maps are received, agent A makes a decision, taking into account the incurred costs for itself and for its immediate neighbours. As long as there is a unique minimizer, this schedule will be assigned, and the agent will inform its neighbours.

If there is no unique minimizer, the decision is delayed until more information is provided, i.e. until a neighbours has picked a schedule. If all decisions are delayed, this uniqueness constraint is relaxed. Additionally, in order to prevent race conditions, if – since the start of an information gathering loop – a neighbour informs A that an assignment has been made, the process is restarted. This ensures that the most up-to-date information is always taken into account, and prevents any two neighbours from picking the same schedule.

Centralized Model-Free Reinforcement Learning Approach (RL)

As an alternative to the DCOP approach – where one needs to define utilities manually to model how the truck planning problem needs to be solved – one can also consider how these agents can learn to solve the problem themselves. RL enables agents to automatically learn what the optimal scheduling decisions are, based on previous experiences without requiring extensive modelling of the utility of a resulting planning. In addition, new 'intelligent' behaviour, not considered yet by human planners, may arise.

Due to the episodic nature of the optimization problem, RL has been applied via the use of a DQN as described in de Vries (2021). Table 1 lists the used constraints. When a truck needs to make a decision, the state information of all orders and other trucks is included in the input of the DQN. A truck then iterates over the list of available orders to calculate the quality (Q-value) of transporting each order. This Q-value represents how valuable an action is (e.g., transport order, wait, chassis change) based on the expected future rewards obtained, and is given in Eq. (4). In this setting, a negative reward will be given to trucks when the chosen order is finished late. Minimizing the reward leads to the least total lateness.

$$Q(s_t, a_t) = \frac{R(\gamma^k - 1)}{k(\gamma - 1)} + \gamma^k V(s_{t+k}) \tag{4}$$

where:

$Q(s_t, a_t)$ = Q-value of action a in state s at time t
k = Number of timesteps to consider
γ = Discount factor
R = Reward when doing action a_t in state s_t
$V(s_{t+k})$ = Value at state s_{t+k} defined as $\overset{max}{a} Q(s_t, a_t)$

The model is trained centrally on a historical dataset spanning 4 years totalling roughly 250,000 orders. To obtain experience during training, actions are chosen via an ϵ-greedy mechanism. When a truck needs to choose an order, it does so randomly with chance ϵ (starting at 0.9 and decreasing over time). Alternatively, it chooses an order based on the policy network. Execution was done centrally in our experiments, but can also take place locally in a decentralized manner, provided that truck state information is shared between the truck agents.

Due to the differences in method execution (specifically, the need for a training phase), a second simulator was built to train the DQN. This simulator generates a random set of orders and trucks for a day from a distribution modelled after historical data. Order characteristics, such as pickup location, destination, and time windows were also varied.

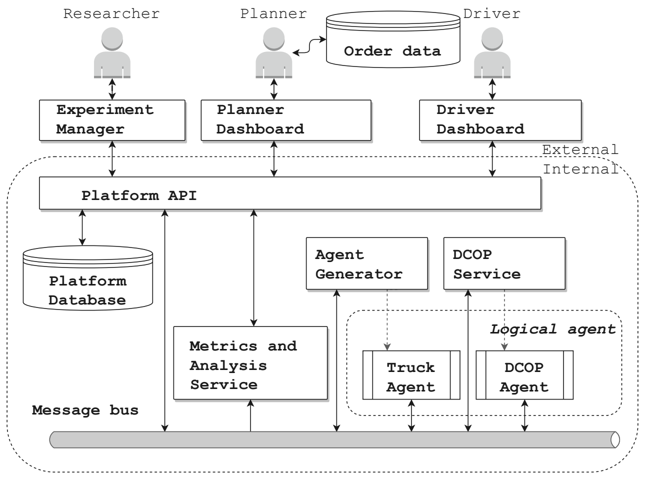

Figure 1: Talking Trucks system architecture.

System Overview and Implementation

As shown in Figure 1, the implemented system is comprised of multiple components, each running as containerized applications. The following main constructs are employed by these components to reason about the planning problem: an *Order*, consisting of a booking identifier and container characteristics – Twenty-foot Equivalent Unit (TEU), weight, etc. – which represents the main unit to be allocated to vehicles for transportation; a *Route stop*, which is associated with an order and holds specific location information and pickup windows; a *Sequence*, which is a list of sequential orders; an *Assignment*, which assigns a specific agent to a sequence and contains attributes to (in)validate or rate it; and finally, a *Planning*, which is a list of assignments.

Central in this architecture is a distributed message bus which is used by virtual agents to communicate preferences for picking up particular orders. Agents react on new orders which become available to them. These are published through an API to the message bus (either manually or via the planner dashboard). Agents listen to this bus for information regarding new orders, updates to orders, and updates to assignments.

Additionally, the message bus is used to inform the managing components to apply updates to the agents they are monitoring. The distributed nature of the message bus allows agents to autonomously react on incoming messages (or be offline for extended periods of time). Furthermore, this design is highly scalable, easily converted into operational software, and allows for individual differences to internal agent logic in a multi-fleet scenario.

A logical agent – the virtual representation (a digital twin) of a truck in the fleet – consists of two components running in parallel: a truck agent implemented in Python to generate locally optimal sequences of orders, and a DCOP agent using an existing CoCoA Java implementation. The truck agent is responsible for executing scheduling operations and scheduling communication with the DCOP-algorithm. Truck agents subscribe to the main topic on the message bus, where new orders are published for a given experiment, and, after calculating their individual loss for a sequence as described in Section , publish that back to the topic for other agents interested in the same orders to see. Truck agents also react to any order updates relevant to them. During a scheduling round, the corresponding DCOP agent is informed about these loss messages and, after solving the assignment problem of trucks to sequences, publishes these solutions back on the bus. Both types of agents are managed by control components, the agent generator and the DCOP service, which are responsible for creation, updating, and removal of individual agents, respectively.

Several User Interface (UI) components communicate with the system backend via the API. Notably, the planner dashboard UI is used by the human planner to add new daily order information to the system; view the generated planning resulting from negotiation between agents; and accept or reject, and provide feedback on specific assignments or the overall planning. The system is backed by a single relational database. Note therefore that, while planning is done decentrally by the agents, the experimentation platform including API and UI components are set up in a centralized manner. By replacing the current distributed message bus with a fully decentralized alternative, and moving to a database-per-stakeholder model, this setup is easily extended to a completely decentralized multi-fleet environment. Finally, an analysis service listens to the message bus for agent and planning messages and, when triggered through the API, will calculate KPIs relating to truck capacity utilisation. Full KPI details are presented in Section .

We are working to provide this cloud-native simulation framework – including agent logic – as open-source tooling available to the academic SOL community under a non-commercial license to further research in the SOL domain. It is currently available on request.

Experimental Design

Experimental data was gathered during three separate experiments at a multi-modal logistics service provider in the Netherlands, taking into account all regional orders scheduled for the same day; Table 2 summarizes. This dataset includes order data – comprised of an identifier, location code, stop type (pickup or delivery), and time window information – and truck data – comprised of an identifier, truck type (regional or port), emission Euro norm, cost of driving 1km, cost of driving an hour, maximum allowed TEU, maximum allowed order weight, and working hours – and is publicly available in pseudonymized form at: https://zenodo.org/record/5777315.

Notably, for Experiment 1, all order time windows were strict (15 minutes), whereas for the Exp. 3, 12 of 41 total orders had large time windows, spanning the full length of the day (up to 12 hours). This type of flexibility has pronounced repercussions for the generation of a daily schedule, separating Exp. 3 from the other two other experiments. This means that the SOLV method, which does not apply time windows but rather employs fixed deadlines (Section), has a considerable disadvantage in this specific experiment. Truck characteristics also vary over the experiments, with Exp. 1 and 2 containing similar truck types, while Exp. 3 has trucks that

Exp.	#	# Orders		
No.	Trucks	Live handling	Decoupling	Total
1	8	22 (57.9%)	16 (42.1%)	38
2	9	15 (40.5%)	22 (59.5%)	37
3	9	30 (73.2%)	11 (26.8%)	41

Table 2: Truck and order statistics in the three experiments.

are more complex, e.g., in working hours that are more heterogeneous and, in part, constrained.

Any particulars during experiments, such as notable cargo not included in the input data, were annotated. The MILP-based solver, different from other methods, also explicitly takes into account mandated 15- and 30-minute breaks for drivers. Other methods do not, as domain experts noted that these typically take place during handling and are therefore largely already present in the dataset.

During the experiments, the full platform ran on a single machine equipped with a 6-core AMD Opteron 6328 CPU and 16GiB RAM. A system consisting of a 12-core Intel Xeon E5645 CPU and 24GiB RAM was used to train the DQN-based model over the course of 24 hours, with observed signs of convergence after 2 hours. Time to generate a solution was in the order of milliseconds for the trained DQN, in the order of seconds for the AGT-DCOP method, and in the order of milliseconds for the linear programming-based solver. Human planners were observed to provide solutions between 30–60 minutes.

After each field trial, the resulting planning was reviewed together with human planners and comments were used to improve data-acquisition (e.g., in the case of adjusting time windows to better reflect real-world situations) and tune the AGT-DCOP and RL methods. To reduce potential bias, there was no interaction with human planners during execution. Notable changes during this phase include the changing of the time windows (when indicated that some were more flexible in actuality than on paper), and the addition of a factor in the utility function to favour orders that should occur earlier on the day (for situations with day-long time windows).

Finally, all baseline methods and final iterations of the AGT-DCOP and RL method were run on the full 3-day dataset. The RAND, EAR-D, and RL methods were each run for 100 iterations because of their stochastic components. We report the mean and standard deviation of these runs. The AGT-DCOP was instantiated with the following parameter values: n: number of schedules to consider for DCOP: 100; w_o: weight factor for number of bookings: 0.5; w_d: weight factor for deadline utility: 1.5; w_e: weight factor for empty driving time: 0.0002. Real world KPIs important to the logistics carrier (including derivatives of methods' explicit objective functions) were automatically generated to output general descriptive metrics (percentage of total orders planned, number of rides planned, and percentage of ordered TEU planned), punctuality metrics (percentage of orders planned early, on time, and late respectively), labour metrics (active, waiting, driving, handling, and working hours), and distance metrics (planned kilometres driven, and percentage driven with load or empty).

Orders are marked as late when the arrival time exceeds the time window deadline by more than 15 minutes; or marked as early when a truck arrives on location 15 minutes or more before the designated time window starts. However, as noted, the SOLV method only uses fixed deadlines and does not incorporate time windows, thereby not considering orders that exceed that deadline.

Late orders are undesirable due to high costs associated with rescheduling a time slot. Early orders do not occur for automated orders, because in these cases, trucks wait just before the terminal (which is reflected in waiting hours). This is not ideal, but exceedingly more desirable than being late. Because the HUMAN method is based on real-world data, it can happen that an early arrival will result in early departure as well, freeing up time. For automated methods, this does not occur because they are never early.

Low active hours during execution are ideal, meaning that all trucks are occupied and actively delivering orders. Start of day waiting hours indicate that trucks can be allocated to perform tasks at the terminal, whereas end of day waiting hours indicate that these trucks can start to pick up new orders. Negative waiting hours indicate overtime for existing trucks and signal a need for additional truck capacity.

Lacking access to truck GPS data, driving distances were computed by taking the great circle distance between the order's origin and destination and multiplying by an expert-supplied factor of 1.2. Driving hours are then determined by multiplying that figure by an estimate of the average driving speed dependent on the type of road (20 km/h for urban roads, 50 km/h for rural roads, and 70 km/h for highways). These figures were found to be representative after manual investigation by domain experts of the logistics service provider when comparing to actual driving times. For the SOLV method, driving times are calculated using a combination of Google Maps, historical data and a slack factor. The human-generated plannings and related metrics for timeliness were annotated manually by domain experts at the logistics carrier based on actual operational information.

Results and Analysis

Figure 2 shows the results on cumulative lateness in minutes for all orders for all three experiments. The AGT-DCOP method outperforms all methods on punctuality (by design, similar to the SOLV method, it will not plan any late orders). The method is able to successfully schedule all orders. Crucially, it was the only method that was able to generate a planning that required 1 truck less than the other methods during Exp. 2.

The HUMAN method performs almost as well, generating very punctual allocations, but results in a number of extreme cases at both the early and late tail of the distribution. Results of the EAR-D heuristic are very similar to the RL method, with the simple EAR-D method working surprisingly well for homogeneous orders (common in Exp. 3) and overall just outperforming the latter. Both methods produce more slightly delayed orders than the HUMAN method. The SOLV method is able to schedule 94.7% of 38 orders in

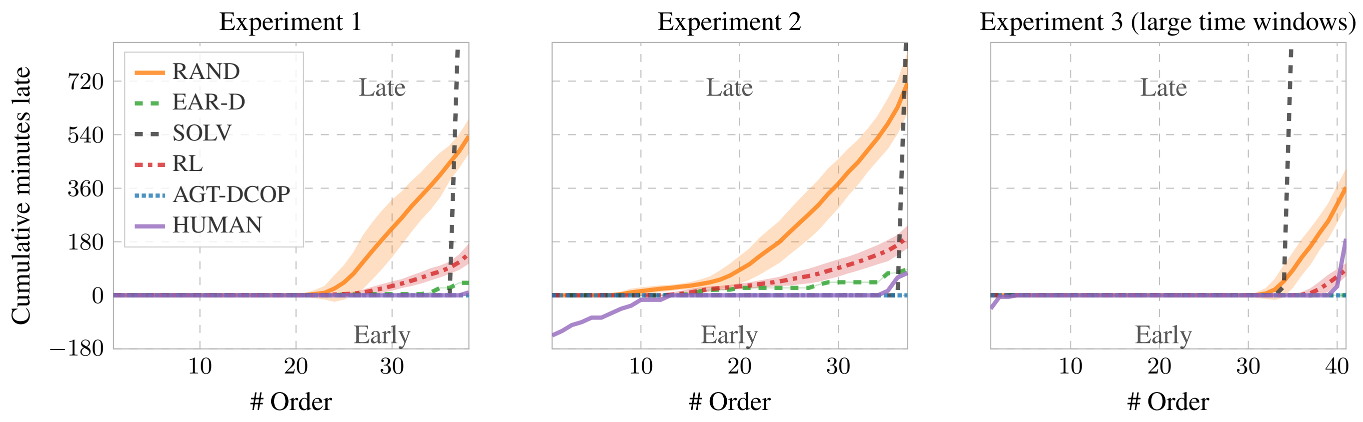

Figure 2: Order lateness in minutes, per planning method. Orders are sorted in ascending order or lateness for each experiment. Bands for RAND, EAR-D, and RL methods show standard deviation for the 100 iterations.

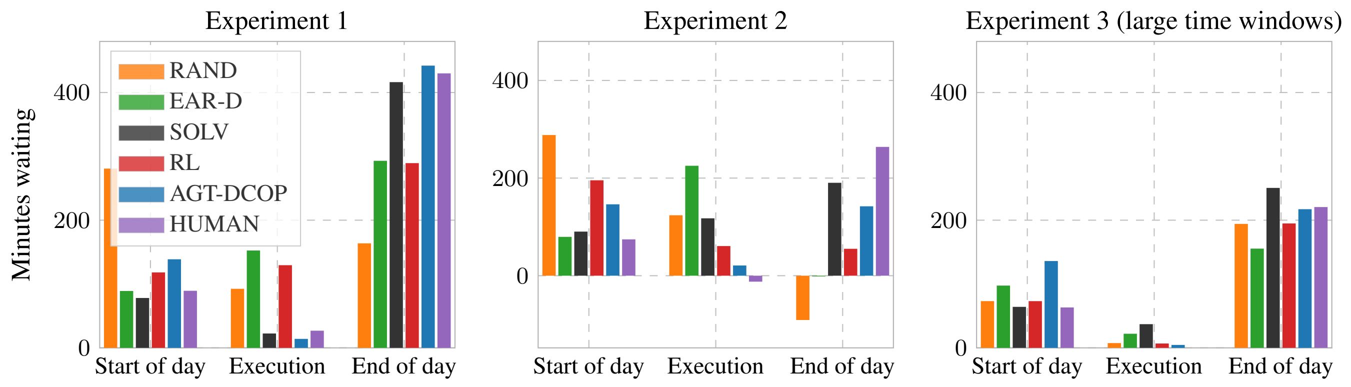

Figure 3: Average combined waiting time for trucks in minutes, at the start of the day, during execution, and at the end of the day, per planning method.

Exp. 1, 97.3% of 37 orders in Exp. 2, and 82.9% of 41 orders in Exp. 3. By design it will not plan orders that are late, therefore any non-scheduled orders are assigned maximum lateness. As expected, the RAND method clearly results in the highest number of late orders.

Considering truck waiting hours shown in Figure 3, we observe that both our AGT-DCOP and the HUMAN method outperform all other methods for waiting time during execution, averaging a maximum waiting time over all three days of 13.19 ($\sigma = 6.71$) and 13.03 ($\sigma = 10.99$) minutes respectively. In Exp. 2, the HUMAN method is shown to outperform all other methods by roughly 2 end-of-day waiting hours. This can be attributed to the fact that the AGT-DCOP method completes the planning with 1 truck fewer. The RL method outperforms both heuristics on average for both waiting hours incurred during execution and those available at end-of-day. The SOLV method performs well on the first two experiments, but because it does not consider time windows and only maintains strict deadlines, it cannot capitalize on that in Exp. 3. Again, the RAND method is distinctly the worst-performing method. We observe that it requires a lot of overtime during Exp. 2 as indicated by a negative number of waiting hours.

Regarding average kilometres driven, we found that all methods are very much aligned. Although routes to delivery locations do not differ between the EAR-D, RAND, RL, and AGT-DCOP methods, we observed that when comparing against the HUMAN method there is less than a 5% (or 50 km) difference in average kilometres driven per truck, with the HUMAN method obtaining the lowest number of driven kilometres which we attribute to the fact that this method directly monitors truck GPS locations. (We do not compare driven kilometres for the SOLV method as that method enforces empty return trips to the terminal which other methods do not require.) Costs for a logistics operator are determined by number of driven kilometres and waiting hours which involves driver salaries and taxes, but moreover, costs for not meeting a delivery deadline, and having to charter another truck, are even more critical. In this context, AGT-DCOP performs best due to needing 1 truck less to complete the planning during Exp. 2.

When reviewing the final generated plannings of the AGT-DCOP method with human experts, it was observed that the plans were feasible and realistic for execution.

Method	Decentralized	Scalability	Punctuality	Modelling effort	Data requirements	Run time (range)
HUMAN	✗	Low	High	–	Medium	Hour
RAND	✓	High	Low	Low	Low	Milliseconds
EAR-D	✓	High	Medium	Low	Low	Milliseconds
SOLV	✗	Medium	High*	Medium	Medium	Milliseconds
AGT-DCOP	✓	High	High	High	Medium	Seconds
RL	✗ (possible)	Medium	Medium	Medium	High	Milliseconds (inference)

Table 3: Comparative strengths and weaknesses. *The (SOLV) method is punctual for orders that it considers because it will not plan any late orders, but is therefore not able to schedule all orders, whereas all other methods are.

Table 3 presents the main strengths and weaknesses of methods used in this work. Human planning is punctual, but also costly. Simple heuristics (e.g., EAR-D) scale well but provide sub-optimal results, whereas the centralized MILP-based SOLV method is punctual by design but fails to schedule all orders. RL is more data dependent than other methods and found to be more applicable to more complex scenarios. The AGT-DCOP method requires significant modelling effort, but generates very punctual plannings and scales well.

Conclusion and Future Work

This work considered the order scheduling problem using real-world data, and contrasted a decentralized model-based approach to a centralized, but model-free approach, benchmarking these methods against human experts, a MILP-based solver, and two heuristics. These methods were implemented in a cloud-native simulation framework with a low barrier to operational deployment, automatically incorporating human feedback.

Results on real-world data show that it is realistic to support human planners in the logistics domain with a decentralized SOL solution. The presented AGT-DCOP method is able to outperform both human expert and heuristic baselines on punctuality, delivering all orders on time, and maintaining truck activity similar to the human expert baseline. The centralized RL method, optimized on maximizing punctuality rather than maximizing number of orders delivered, does not score better than human expert planning in terms of average lateness and induces a higher cost on truck waiting time than the AGT-DCOP method. While the RL method does not outperform the greedy EAR-D heuristic on average lateness, its planning results are much more efficient in terms of truck activity. RL's strengths lie in its ability to model complex dynamics in situations where it is infeasible to manually construct utility functions, and thus environments where a model-based approach cannot be implemented.

There are benefits to automated decentralized planning: greater scalability and improved data security, and the ability to compute with a large number of constraints and a highly heterogeneous group of actors. However, we also note several limitations on these methods. There is increased complexity in managing distributed software components, especially in trying to capture a noisy and uncertain world into neat constraints and assignments. Second, human planners can currently deal with a much larger range of operational changes and issues (e.g., when assigning two trucks to a single driver for repeated live-handling orders), some of which are notoriously difficult to model. It is critical to involve planners in the design process of automated tools. Care should be taken that human values are not lost in the emergence of automated control. Third, human decision makers insist on algorithmic decisions being explainable, which can be challenging due to (sometimes obfuscated) utility functions. We found that while individual truck plannings can be explained well to operators, doing this for a full planning is more difficult when peculiarities or novel solutions are found by the algorithm. Fourth, a form of oversight should be in place when constructing automated decentralized coordination environments to keep actors behaving in good faith.

Future work includes evaluating our models on larger and more complex problems (e.g., involving anomalies) where the RL method's strengths are more apparent; a closer investigation of the multi-objective Pareto frontier to better evaluate the cost of an improvement in punctuality; implementing a more advanced search through the sequence space; and adding support for federated multi-stakeholder environments. Additionally, there exist various strategies to decentralize the proposed RL method, for example through federated learning, mitigating some of its disadvantages.

Regarding future steps for SOL, we believe that as automation and machine learning in the field continues to proliferate in order to support growing shipment volumes and available sensor-data, the demand for explainability for human operators will be of the highest importance in order to trust automated decision support solutions.

Acknowledgements

This work is part of the TNO 'Talking Trucks' project and is supported by Van Berkel Logistics, DHL Global Forwarding and SmartPort Rotterdam. It has received funding from the TNO Digital Twin lifecycle technology programme for Reliable Digital Twinning. This research was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under grant number 952215. The authors thank the anonymous reviewers, and Paolo Pileggi and Lóránt Tavasszy for many helpful comments that improved the quality of this manuscript.

References

Baldacci, R.; Mingozzi, A.; and Roberti, R. 2012. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European J. of Operational Research*, 218(1): 1–6.

Berndt, J. O. 2011. Self-Organizing Logistics Process Control: An Agent-Based Approach. In *Proc. ICAART*, 397–412.

Chao, I.-M. 2002. A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, 29(1): 33–51.

Dantzig, G. B.; and Ramser, J. H. 1959. The Truck Dispatching Problem. *Management Science*, 6(1): 80–91.

de Vries, Y. C. 2021. *Reinforcement learning for order distribution in self-organizing logistics*. Master's thesis, Delft University of Technology.

Feng, F.; Pang, Y.; Lodewijks, G.; and Li, W. 2017. Collaborative framework of an intelligent agent system for efficient logistics transport planning. *Computers & Industrial Engineering*, 112: 551–567.

Gerrits, B. 2020. Towards a Unifying Framework for Self-Organization in Transport Logistics. In *Proc. ICCL*.

Gombolay, M. C.; Jensen, R.; Stigile, J.; Golen, T.; Shah, N.; Son, S.; and Shah, J. A. 2018. Human-Machine Collaborative Optimization via Apprenticeship Scheduling. *J. Artificial Intelligence Research*, 63: 1–49.

Grinshpoun, T.; Grubshtein, A.; Zivan, R.; Netzer, A.; and Meisels, A. 2013. Asymmetric Distributed Constraint Optimization Problems. *Journal of Artificial Intelligence*, 47: 613–647.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proc. ICML*, 1861–1870. PMLR.

Hirayama, K.; and Yokoo, M. 1997. Distributed partial constraint satisfaction problem. In *Principles and Practice of Constraint Programming*, 222–236.

Irannezhad, E.; Prato, C. G.; and Hickman, M. 2020. An intelligent decision support system prototype for hinterland port logistics. *Decision Support Systems*, 130: 113227.

Joe, W.; and Lau, H. C. 2020. Deep Reinforcement Learning Approach to Solve Dynamic Vehicle Routing Problem with Stochastic Customers. In *Proc. ICAPS*, 394–402.

Karunakaran, J. 2020. *Optimal fleet assignment in inland container logistics*. Master's thesis, Eindhoven University of Technology.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. A. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR*, abs/1312.5602.

Nguyen, T. T.; Nguyen, N. D.; and Nahavandi, S. 2020. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Trans. Cybern.*, 50(9): 3826–3839.

Pan, S.; Trentesaux, D.; and Sallez, Y. 2016. Specifying Self-organising Logistics System: Openness, Intelligence, and Decentralised Control. In *International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing*, 93–102. Springer.

Quak, H.; van Kempen, E.; and Hopman, M. 2018. Moving towards practical implementation of self-organizing logistics – making small steps in realizing the PI vision by raising awareness. *International Physical Internet Congres*, 106–119.

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

Song, X.; Jones, D.; Asgari, N.; and Pigden, T. 2020. Multi-objective vehicle routing and loading with time window constraints: a real-life application. *Annals of Operations Research*, 291: 799–825.

Tang, H.; Wang, A.; Xue, F.; Yang, J.; and Cao, Y. 2021. A novel hierarchical soft actor-critic algorithm for multi-logistics robots task allocation. *IEEE Access*, 9: 42568–42582.

ter Mors, A.; Zutt, J.; and Witteveen, C. 2007. Context-Aware Logistic Routing and Scheduling. In *Proc. ICAPS*, 328–335.

van Leeuwen, C. J.; and Pawełczak, P. 2017. CoCoA: A Non-iterative Approach to a Local Search (A)DCOP Solver. In *Proc. AAAI*, 3944–3950.

van Ommeren, C. R.; Fransen, R. W.; Pingen, G. L. J.; van Leeuwen, C. J.; Paardekoper, J. P.; and van Meijeren, J. C. 2020. Letting Digital Twins Run the Show: Exploring possibilities of letting vehicles plan and organise transportation themselves. Technical report, TNO.

Yeoh, W. 2018. Towards Improving the Expressivity and Scalability of Distributed Constraint Optimization Problems. In *IJCAI*, 5734–5738.