Towards systematic evaluation of multi-agent systems in large scale and dynamic logistics[⋆]

Rinde R.S. van Lon and Tom Holvoet

iMinds-DistriNet, KU Leuven, 3001 Leuven, Belgium.
Rinde.vanLon@cs.kuleuven.be, Tom.Holvoet@cs.kuleuven.be

Abstract. A common hypothesis in multi-agent systems (MAS) literature is that decentralized MAS are better at coping with dynamic and large scale problems compared to centralized algorithms. Existing work investigates this hypothesis in a limited way, often with no support for further evaluation, slowing down the advance of more general conclusions. Investigating this hypothesis more systematically is time consuming as it requires four main components: 1) formal metrics for the variables of interest, 2) a problem instance generator using these metrics, 3) (de)centralized algorithms and 4) a simulation platform that facilitates the execution of these algorithms. Present paper describes the construction of an instance generator based on previously established formal metrics and simulation platform with support for (de)centralized algorithms. Using our instance generator, a benchmark logistics dataset with varying levels of dynamism and scale is created and we demonstrate how it can be used for systematically evaluating MAS and centralized algorithms in our simulator. This benchmark dataset is essential for enabling the adoption of a more thorough and systematic evaluation methodology, allowing increased insight in the strengths and weaknesses of both the MAS paradigm and operational research methods.

1 Introduction

In pickup and delivery problems (PDPs) a fleet of vehicles is tasked with transporting customers or goods from origin to destination [23,27]. In dynamic PDPs the orders describing the vehicles' tasks arrive during the operating hours [1], necessitating online assignment of vehicles to orders. The dynamic nature and potential large scale of this problem makes exact algorithms often infeasible.

Decentralized multi-agent systems (MASs) are often presented as a good alternative to centralized algorithms [3,6,29], MASs are especially promising for large scale and dynamic problems due to their ability to make quick local decisions. Previous work has shown that MASs can sometimes outperform centralized algorithms in specific cases [3,18,19,20]. However, to the best of our knowledge there has never been a systematic effort to compare centralized algorithms to decentralized MASs with varying levels of dynamism and scale.

[⋆] The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-25524-8_16. This is a postprint version.

Although the previously mentioned papers each do a thorough evaluation of a MAS applied to a logistics problem, it is often hard to do further comparisons using these papers because of the lack of available problem data, source code or both. It has been argued before that this is a problem in science in general [8], and in multi-agent systems literature in particular [16].

In this paper we introduce a dataset generator and a benchmark dataset of the dynamic pickup and delivery problem with time windows (PDPTW) with support for varying three variables. The degree of dynamism and urgency of a dynamic PDPTW are two variables that were introduced before [14]. The proposed dataset contains an additional variable, scale, that we define in the context of PDPTW as a multiplier applied to the number of vehicles and orders in a problem. Using this dataset it will be possible to systematically investigate the following hypotheses in the context of PDPTW:

– Multi-agent systems perform better when compared to centralized algorithms on very dynamic problem instances
– Multi-agent systems perform better when compared to centralized algorithms on more urgent problem instances
– Multi-agent systems perform better when compared to centralized algorithms on large scale problem instances

Investigating these hypotheses should lead to insight in the performance of both decentralized MASs and centralized algorithms for PDPTWs. These insights can then be used to make more informed decisions when designing a system that needs to cope with dynamic, urgent and large scale problems. Additionally, the dataset generator, the benchmark dataset instance and the simulator [15] that we use are open sourced. This improves the reproducibility of the current paper while presenting an opportunity for other researchers to investigate the above hypotheses using their own algorithms.

The paper is organized as follows. First, the relevant literature is discussed (Section 2) and we define dynamic PDPTWs including the measures for dynamism, urgency and scale and the measure for algorithm performance (Section 3). This is followed by a description of the dataset generator and dataset benchmark instance (Section 4). It is demonstrated how the hypotheses of dynamism, urgency and scale can be investigated using the proposed benchmark instance (Section 5), leading to the conclusion that the benchmark dataset facilitates a systematic and long term research effort into these hypotheses (Section 6).

2 Related work

Several literature surveys discuss the dynamic vehicle routing problem (VRP) and its special case, dynamic PDPTW [1,5,24,26]. The dynamic PDPTW is often treated as a stochastic problem where some a priori information is known about the orders. This section only discusses papers that do not use a priori information but view the problem from a completely dynamic perspective.

2.1 Centralized algorithms

Madsen et al. [17] developed an insertion heuristic for the dynamic dial-a-ride-problem (DARP) with time windows for moving elderly and disabled people. Potvin et al. [25] presented a learning system based on linear programming that can learn an optimal policy taking into account decisions of an expert in past scenarios. Mitrović-Minić et al. [22] presented an approach based on two time horizons: a short time horizon aimed at achieving the short-term goal of minimization of distance traveled, and a longer time horizon aimed at achieving the long-term goal of facilitating the insertion of future requests. Gendreau et al. [4] introduced a dynamic version of tabu search with a neighboring structure based on ejection chains. When new requests arrive, the algorithm reacts by insertion and ejection moves and with local search.

2.2 Multi-agent systems

An alternative approach to the dynamic PDPTW is using a decentralized MAS instead of a centralized planner. Fischer et al. [3] used a MAS with the extended contract net protocol for cooperative transportation scheduling and they showed that its performance was comparable to existing operational research (OR) techniques. Mes et al. [20] compared traditional heuristics with a distributed MAS that uses a Vickrey auction to bid for new pickup and delivery requests when they appear, showing that the MAS approach performs often better than traditional heuristics. In subsequent work Mes and Van der Heijden [21] further improved the performance of the MAS by introducing a look-ahead mechanism in which bidding uses value functions to estimate the expected future revenue of inserting a new order in an agent plan. Máhr et al. [19] thoroughly evaluated a MAS with auctions and a mixed-integer program on real world data of a PDPTW. Their results show that both approaches have comparable performance. Glaschenko et al. [6] discussed the deployment of a MAS for a taxi company in London, adopting the MAS led to an increase of taxi fleet utilization by 5 - 7 %.

3 Dynamic pickup-and-delivery problems

We base our definition of dynamic PDPs on [14] which is an adaptation of the definition of [4]. In PDPs there is a fleet of vehicles responsible for the pickup-and-delivery of items. The dynamic PDP is an online problem, the customer transportation requests are revealed over time during the fleet's operating hours. It is further assumed that the fleet of vehicles has no prior knowledge about the total number of requests nor about their locations or time windows.

3.1 Formal definition

For describing the dynamic PDP we adopt the formal definition of [14]. A *scenario*, which describes the unfolding of a dynamic PDP, is defined as a tuple:

$$\langle \mathcal{T}, \mathcal{E}, \mathcal{V} \rangle := \text{scenario},$$

where

$$\begin{aligned} [0, \mathcal{T}) &:= \text{time frame of the scenario,} & \mathcal{T} > 0 \\ \mathcal{E} &:= \text{list of events,} & |\mathcal{E}| \geq 2 \\ \mathcal{V} &:= \text{set of vehicles,} & |\mathcal{V}| \geq 1 \end{aligned}$$

$[0, \mathcal{T})$ is the period in which the fleet of vehicles $\mathcal{V}$ has to handle all customer requests. The events represent customer transportation requests. Since we consider the purely dynamic PDPTW, all events are revealed between time 0 and time $\mathcal{T}$. Each event $e_i \in \mathcal{E}$ is defined by the following variables:

$$\begin{aligned} a_i &:= \text{announce time} \\ p_i &:= [p_i^L, p_i^R) = \text{pickup time window, } p_i^L < p_i^R \\ d_i &:= [d_i^L, d_i^R) = \text{delivery time window, } d_i^L < d_i^R \\ pst_i &:= \text{pickup service time span} \\ dst_i &:= \text{delivery service time span} \\ ploc_i &:= \text{pickup location} \\ dloc_i &:= \text{delivery location} \\ tt_i &:= \text{travel time from pickup location to delivery location} \end{aligned}$$

Reaction time is defined as:

$$r_i := p_i^R - a_i = \text{reaction time} \tag{1}$$

The time window related variables of a transportation request are visualized in Figure 1.

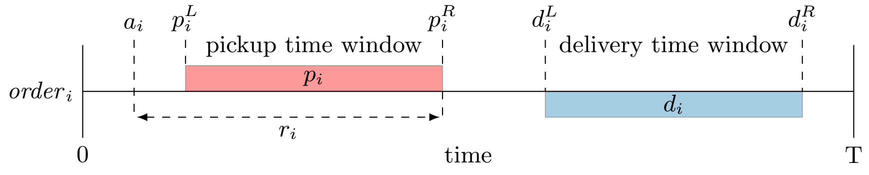


Fig. 1: Visualization of the time related variables of a single order event $e_i \in \mathcal{E}$.

Furthermore we assume that:

- vehicles start at a depot and have to return after all orders are handled;
- the fleet of vehicles $\mathcal{V}$ is homogeneous;
- the cargo capacity of vehicles is infinite (e.g. courier service);
- the vehicle is either stationary or driving at a constant speed;
- vehicle diversion is allowed, this means that a vehicle is allowed to divert from its destination at any time;

– vehicle fuel is infinite and driver fatigue is not an issue;
– the scenario is completed when all pickup and deliveries have been made and all vehicles have returned to the depot; and,
– each location can be reached from any other location.

Vehicle schedules are subject to both hard and soft constraints. The opening of time windows is a hard constraint, hence vehicles need to adhere to these:

$$sp_{ij} \geq p_i^L \tag{2}$$

$$sd_{ij} \geq d_i^L \tag{3}$$

Here, sp_{ij} is the start of the pickup operation of order event e_i by vehicle v_j; similarly, sd_{ij} is the start of the delivery operation of order event e_i by vehicle v_j. The time window closing (p_i^R and d_i^R) is a soft constraint incorporated into the objective function, it is defined similarly to [4] and needs to be minimized:

$$min := \sum_{j \in \mathcal{V}} (vtt_j + td\{bd_j, \mathcal{T}\}) + \sum_{i \in \mathcal{E}} \left(td\left\{sp_{ij}, p_i^R\right\} + td\left\{sd_{ij}, d_i^R\right\}\right) \tag{4}$$

where

$$td\{\alpha, \beta\} := max\{0, \alpha - \beta\} \ = \ \text{tardiness} \tag{5}$$

Here, vtt_j is the total travel time of vehicle v_j; bd_j is the time at which vehicle v_j is back at the depot. In summary, the objective function computes the total vehicle travel time, the tardiness of vehicles returning to the depot and the total pickup and delivery tardiness.

We further impose the following hard constraints on the construction of scenarios to ensure consistency and feasibility of individual orders:

$$r_i \geq 0 \tag{6}$$

$$d_i^R \geq p_i^R + pst_i + tt_i \tag{7}$$

$$d_i^L \geq p_i^L + pst_i + tt_i \tag{8}$$

These constraints are visualized in Figure 2. The reaction time constraint (eq. 6)

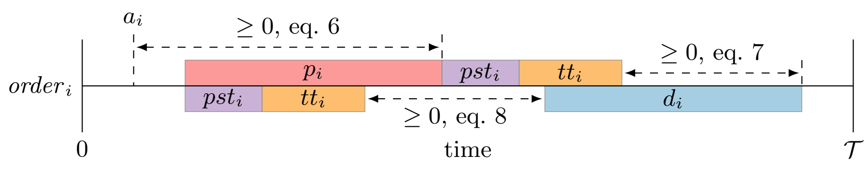


Fig. 2: Visualization of the time window constraints of an order event $e_i \in \mathcal{E}$.

ensures that an order is always announced before its due date. The time window constraints (eq. 7 and eq. 8) ensure that pickup and delivery time windows are compatible with each other. Hence, a pickup operation started at any time within p_i guarantees feasibility of a delivery within d_i given that a vehicle is available and respecting vehicle capacity, service time and travel time constraints.

3.2 Dynamism

In this section we describe the measure for the degree of dynamism first defined in [14]. Informally, a scenario that changes continuously is said to be dynamic while a scenario that changes occasionally is said to be less dynamic. In the context of PDPTWs a change is an event that introduces additional information to the problem, such as the events in $\mathcal{E}$. More formally, the degree of dynamism, or the continuity of change, is defined as:

$$\mathit{dynamism} := 1 - \frac{\sum_{i=0}^{|\varDelta|} \sigma_i}{\sum_{i=0}^{|\varDelta|} \bar{\sigma}_i} \tag{9}$$

where

$$\varDelta := \{\delta_0, \delta_1, \dots, \delta_{|\mathcal{E}|-2}\} = \{a_j - a_i | j = i + 1 \land \forall a_i, a_j \in \mathcal{E}\} \tag{10}$$

$$\theta := \text{perfect interarrival time} = \frac{\mathcal{T}}{|\mathcal{E}|} \tag{11}$$

$$\sigma_i := \begin{cases} \theta - \delta_i & \text{if } i = 0 \text{ and } \delta_i < \theta \\ \theta - \delta_i + \dfrac{\theta - \delta_i}{\theta} \times \sigma_{i-1} & \text{if } i > 0 \text{ and } \delta_i < \theta \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

$$\bar{\sigma}_i := \theta + \begin{cases} \dfrac{\theta - \delta_i}{\theta} \times \sigma_{i-1} & \text{if } i > 0 \text{ and } \delta_i < \theta \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

This measure can compute the degree of dynamism of any scenario.

3.3 Urgency

In [14] urgency is defined as the maximum reaction time available to the fleet of vehicles in order to respond to an incoming order. Or more formally:

$$\mathit{urgency}\,(e_i) := p_i^R - a_i = r_i \tag{14}$$

To obtain the urgency of an entire scenario the mean and standard deviation of the urgency of all orders can be computed.

3.4 Scale

Assigning a scale level to a PDP instance allows to conduct a scalability experiment to investigate the existence of a correlation between the scale of a PDP and the computation time and solution quality of an algorithm.

In the context of computer systems scaling up is defined as maintaining a fixed execution time per task while scaling the workload up in proportion to the number of processors applied to it [7]. Analogously, scaling in the context of PDPs can be defined as maintaining a fixed computation time per order while scaling the workload (number of orders) up in proportion to the number of vehicles in the fleet.

However, there are three factors that limit the usefulness of this definition. First, it is known that PDPTWs are NP-hard [27], therefore an exact algorithm for a PDPTW requires time that is superpolynomial in the input size. Therefore, maintaining a fixed computation time per order when using an exact algorithm is infeasible. When using an anytime algorithm (an algorithm that can be stopped at any moment during its execution to return a valid solution) such as a heuristic, maintaining a fixed computation time per order is trivial, but will likely have an influence on the solution quality.

Second, the previously mentioned notion of urgency influences the amount of available computation time. Within an order's urgency period three activities need to be performed, first a vehicle needs to be selected, then the selected vehicle needs to drive towards the pickup location and it needs to perform the actual pickup operation. The longer the computation of the vehicle selection takes, the less time remains for the driving and picking up.

Third, depending on the degree of dynamism there may be many orders with a small interarrival time. Each order that arrives while a computation takes place forces a premature halt and subsequent restart of the algorithm. Therefore, maintaining a fixed computation time per order is nonsensical for PDPTWs.

For these reasons, we define scaling in PDPTWs as *maintaining a fixed objective value per order while scaling the number of orders up in proportion to the number of vehicles in the fleet.* Using this definition, scaling up a scenario $\langle \mathcal{T}, \mathcal{E}, \mathcal{V} \rangle$ with a factor α will create a new scenario $\langle \mathcal{T}, \mathcal{E}', \mathcal{V}' \rangle$ where $|\mathcal{V}'| = |\mathcal{V}| \cdot \alpha$ and $|\mathcal{E}'| = |\mathcal{E}| \cdot \alpha$. To compute the objective value per order, the global objective value needs to be divided by the number of orders.

4 Dataset

This section describes the construction of the scenario generator that creates scenarios with specific levels of dynamism, urgency and scale. Using the scenario generator a benchmark dataset is constructed.

4.1 Scenario generator

To create a scenario generator capable of generating scenarios with specific levels of scale, dynamism and urgency we adapted the generator developed in [14].

Controlling dynamism of time series Based on [14] we assigned a time series generator method to a specific range of dynamism levels such that the entire range $[0, 1]$ is covered (Table 1).

Table 1: Overview of dynamism ranges and the corresponding time series generator used for generating scenarios in that range.

Dynamism range	**Time series generator**
$[0, .475)$	non-homogeneous Poisson process
$[.475, .575)$	homogeneous Poisson process
$[.575, .675)$	Normal distribution
$[.675, 1]$	Uniform distribution

The non-homogeneous Poisson process that is used for $[0, .475)$ has an intensity function based on a sine wave with the following parameters:

$$\lambda(t) = a \cdot \sin(t \cdot f \cdot 2\pi - \pi \cdot p) + h \tag{15}$$
$$a = 1 \qquad \text{amplitude} \tag{16}$$
$$f = 1 \qquad \text{frequency} \tag{17}$$
$$p \sim \mathcal{U}(0, 1) \qquad \text{phase shift} \tag{18}$$
$$h \sim \mathcal{U}(-.99, 1.5) \qquad \text{height} \tag{19}$$

In order to keep the total number of events constant with different levels of dynamism, the amplitude and height parameters are rescaled such that the total area under the intensity function equals $|\mathcal{E}|$.

For the $[.475, .575)$ range we used the homogeneous Poisson process, with the (constant) intensity function defined as:

$$\lambda(t) = \frac{|\mathcal{E}|}{\mathcal{T}} = 30 \tag{20}$$

The normal distribution for the $[.575, .676)$ range is the truncated normal distribution $\mathcal{N}\left(\frac{\mathcal{T}}{|\mathcal{E}|}, 0.04\right)$ with a lower bound of 0 and a standard deviation of 0.04. If a value x was drawn such that $x < 0$, a new number was drawn from the distribution. Truncating a normal distribution actually shifts the mean, hence the mean was rescaled to make sure the effective mean was equal to $\frac{\mathcal{T}}{|\mathcal{E}|}$.

In the $[.675, 1]$ range a uniform distribution with mean $\frac{\mathcal{T}}{|\mathcal{E}|}$ and a maximum deviation from the mean, σ, is used. The σ value is (for each scenario again) drawn from the truncated normal distribution $\mathcal{N}(1, 1)$ with bounds $[0, 15]$. If a value σ is obtained from the distribution such that $\sigma > 15$ or $\sigma < 0$ a new value is drawn. Since the mean is not scaled, the effective mean of σ is higher than 1.

Generating comparable scenarios with different dynamism, urgency and scale levels The generator should be able to generate a set of scenarios where all settings are the same except for dynamism, urgency and scale levels. Also, any interactions between variables should be minimized, e.g. dynamism should not correlate with time window intervals. This ensures that any effect measured is solely caused by the difference in dynamism, urgency and or scale.

Because the dataset generator is stochastic, the number of events $|\mathcal{E}|$ and the degree of dynamism of a scenario can not be directly controlled. To construct a consistent dataset, scenarios that do not have exactly $|\mathcal{E}|$ events are rejected. For each desired dynamism level a bin with an acceptable deviation is defined, only generated scenarios with a dynamism value that lies within a bin are accepted.

We further define the concept of *office hours* as the period $[0, \mathcal{O})$ in which new orders are accepted. To ensure feasibility of individual orders we need to take into account the travel time, service time durations and urgency:

$$\mathcal{O} = \mathcal{T} - pst_{max} - dst_{max} - \begin{cases} 2 \cdot tt_{max} & \text{if } u < \frac{1}{2} \cdot tt_{max} \\ 1\dfrac{1}{2} \cdot tt_{max} - u & \text{otherwise} \end{cases} \tag{21}$$

Here, pst_{max} and dst_{max} are the maximum pickup and delivery service times respectively, tt_{max} is the maximum travel time between a pickup and delivery location, and u is urgency.

The pickup and delivery time windows have to be randomly chosen while respecting the constraints as set out by the urgency level and the announce time. The p_i^R is defined as the sum of a_i and u, hence it follows that p_i^L needs to be between a_i and the sum of a_i and u:

$$p_i^L = \begin{cases} \sim \mathcal{U}\big(a_i, p_i^R - 10\big) & \text{if } u > 10 \\ a_i & \text{otherwise} \end{cases} \tag{22}$$

Here, 10 is the minimum pickup time window length unless urgency is less than 10, in that case the urgency level equals the pickup time window length. The upper bound of d_i^R can be defined as:

$$ubd_i^R = \mathcal{T} - tt(dloc_i, depot_{loc}) - dst_i \tag{23}$$

This translates as the latest possible time to start the delivery operation such that the delivery time window constraints are met and the vehicle can still be back at the depot on time. The lower bound of d_i^L was already defined in eq. 8:

$$lbd_i^L = p_i^L + pst_i + tt_i \tag{24}$$

We define a minimum delivery time window length of 10, which then results in an upper bound of d_i^L:

$$ubd_i^L = ubd_i^R - 10 \tag{25}$$

Based on these bounds we draw the opening of the delivery time window from the following uniform distribution:

$$d_i^L \sim \mathcal{U}\big(lbd_i^L, \max\big(lbd_i^L, ubd_i^L\big)\big) \tag{26}$$

To find d_i^R we need to redefine the lower bound (from eq. 7) by using the actual value of d_i^L:

$$lbd_i^R = \min\big(\max\big(p_i^R + pst_i + tt_i, d_i^L + 10\big), ubd_i^R\big) \tag{27}$$

Finally, the closing of the delivery time window is defined as:

$$d_i^R \sim \mathcal{U}\big(lbd_i^R, ubd_i^R\big) \tag{28}$$

For the pickup and delivery service times we choose $pst_i = dst_i = 5$ minutes.

All locations in a scenario are points on the Euclidean plane. It has a size of 10 by 10 kilometer with a depot at the center of this square. Vehicles start at the depot and have a constant travel speed of 50 km/h. All pickup and delivery locations are drawn from a two dimensional uniform distribution $\mathcal{U}_2(0, 10)$.

4.2 Benchmark dataset

The benchmark dataset that we created for this paper has three levels for each of the dimensions of interest resulting in a total of $3 \cdot 3 \cdot 3 = 27$ scenario categories. The dimensions of interest are dynamism, urgency and scale, the used values are listed in Table 2a, the other parameters are listed in Table 2b. Since the generation of the order arrival times is a stochastic process the exact degree of dynamism can not be controlled. Therefore, we define a dynamism bin using a radius of 1% around each dynamism value. For this dataset, we consider a scenario with dynamism d where $b - .01 < d < b + .01$ to have dynamism b, where b is one of the dynamism bins listed in Table 2a.

Table 2: Overview of the parameters used to generate the benchmark dataset.

(a) Dimensions

Dimension	**Values**		
Dynamism	.2	.5	.8
Urgency	5	20	35
Scale	1	5	10

(b) Settings

Parameter	**Value**
$\mathcal{T}$	8 hours
$\lvert\mathcal{E}\rvert$	scale $\cdot$ 240
$\lvert\mathcal{V}\rvert$	scale $\cdot$ 10

For each scenario category 50 instances are generated, resulting in a total of $50 \cdot 27 = 1350$ scenarios. Each scenario is written to a separate file with the following name format: `dynamism-urgency-scale-id.scen`, for example `0.20-5-1.00-0.scen` depicts a scenario with 20% dynamism, an urgency level of 5 minutes, a scale of 1 and id 0. This format allows easy selection of a subset of the dataset. The scenario file contains the entire scenario in JavaScript Object Notation (JSON). Time in a scenario is expressed in milliseconds, distance in kilometer and speed in kilometer per hour. A scenario is considered to be finished when all vehicles are back at the depot and the current time is $\geq \mathcal{T}$.

The open source discrete time simulator RinSim [15] version 4.0.0 [13] has native support for the scenario format. With RinSim it is easy to run the scenario with centralized algorithms and multi-agent systems, allowing researchers to only have to focus on their algorithms. For reproducibility, the code of the dataset generator is released [11] as well as the dataset scenarios [10] and all other code and results [9].

5 Demonstration

As a demonstration a centralized algorithm is compared with a decentralized multi-agent system on 10 instances of each category in the benchmark dataset, resulting in a total of 270 experiments per approach. For reproducibility, the code and results of this experiment are published on an accompanying web page [9]

5.1 Heuristics

Just as in [14] two well known heuristics are used, the cheapest insertion heuristic (Algorithm 1) and the 2-opt optimization procedure (Algorithm 2). Since the 2-opt procedure requires a complete schedule as input, it uses the cheapest insertion heuristic to construct a complete schedule first. Both these algorithms have been used in earlier work for vehicle routing problems [2,28].

```
Input: $\langle \mathcal{T}, \mathcal{E}, \mathcal{V} \rangle$;                          /* A scenario as input */
Data: $\mathcal{S}$;                                       /* the current schedule or $\varnothing$ */
$\mathcal{S}_{\text{best}} = \varnothing$
foreach $e \in \mathcal{E}, e \notin \mathcal{S}$ do
    /* generate all PDP insertion points in the current schedule:            */
    insertions = generate_insertion_points($\mathcal{S}$)
    for $i \in insertions$ do
        /* construct a new schedule by inserting $e$ at insertion $i$        */
        $\mathcal{S}_{\text{new}}$ = construct($\mathcal{S}$,e,i)
        if $cost(\mathcal{S}_{new}) < cost(\mathcal{S}_{best})$ then
            $\mathcal{S}_{\text{best}} = \mathcal{S}_{\text{new}}$
        end
    end
end
```

Algorithm 1: Cheapest insertion heuristic, source code available in [12].

```
Input: $\mathcal{S}$
$\mathcal{S}_{\text{best}} = \mathcal{S}$
swaps = generate_swaps($\mathcal{S}$)
foreach $e \in swaps$ do
    $\mathcal{S}_{\text{new}}$ = swap($\mathcal{S}$,e)
    if $cost(\mathcal{S}_{new}) < cost(\mathcal{S}_{best})$ then
        $\mathcal{S}_{\text{best}} = \mathcal{S}_{\text{new}}$
    end
end
/* If a better schedule has been found, we start another iteration          */
if $\mathcal{S}_{best} \neq \mathcal{S}$ then
    2-opt($\mathcal{S}_{\text{best}}$)
end
```

Algorithm 2: 2-opt procedure, source code available in [12].

5.2 Centralized algorithm

Each time a new order is announced the cheapest insertion heuristic is executed to produce a new schedule for the fleet of vehicles. It is assumed that execution of the algorithm is instantaneous with respect to the dynamics of the simulations.

5.3 Contract net protocol multi-agent system

The multi-agent system implementation is based on the contract net protocol (CNP) as described by Fischer et al. [3]. For each incoming order an auction is organized, when the auction is finished the order will be assigned to exactly one vehicle. All vehicles always bid on each order, the bid contains an estimate of the additional cost that including the new order in the vehicles assignment would incur. This estimate is computed using the cheapest insertion heuristic as described in Algorithm 1. The vehicle with the lowest bid will win the auction and receive the order. Each vehicle computes a route to visit all its pickup and delivery sites using the 2-opt procedure described in Algorithm 2.

5.4 Results and analysis

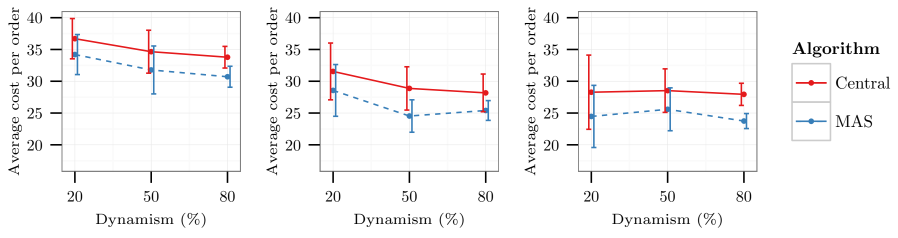
The results[1] of the experiments are plotted along the dynamism, urgency and scale dimension in Figures 3, 4 and 5 respectively. Although all results indicate that the MAS performs better than the centralized algorithm, the current experiment is too limited to verify the hypotheses posed in this paper. Instead, we discuss the behavior of both algorithms with respect to the dimensions of interest.

Dynamism Figure 3 shows that the level of dynamism has very little influence on the performance of both the MAS and the centralized algorithm. This lack of effect is very consistent among all urgency and scale settings.

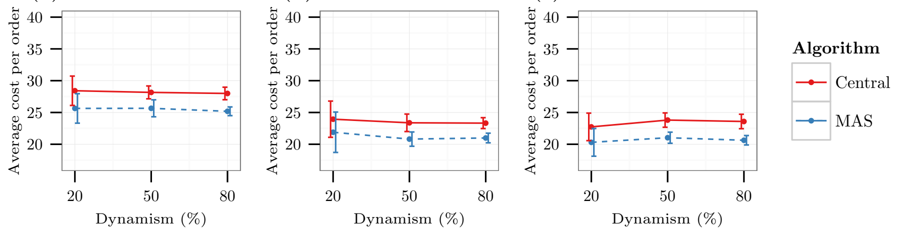
Urgency In Figure 4 a clear trend can be observed for both algorithms, the less urgent orders are, the lower the average cost per order is. This effect can be explained by the fact that when orders are less urgent, vehicles have more time to handle other nearby orders first while still respecting the time windows.

Scale Contrary to what one would expect, Figure 5 shows that the larger scale the problem is the lower the average cost of an order. This surprising result can be explained by the fact that computation time is ignored in our current setup, this means that the algorithms have enough time to deal with greater complexity of larger scale problems. The lower average cost per order can be explained by the fact that with more vehicles the average distance of a new order to the closest vehicle is smaller, resulting in reduced average travel times and tardiness.

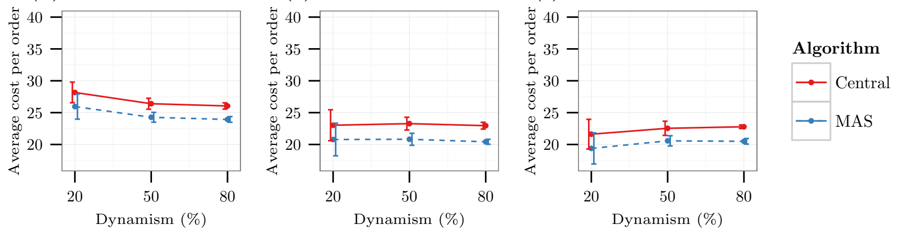
[1] In [9] the raw results are published.



(a) Scale = 1, urg = 5 (b) Scale = 1, urg = 20 (c) Scale = 1, urg = 35



(d) Scale = 5, urg = 5 (e) Scale = 5, urg = 20 (f) Scale = 5, urg = 35



(g) Scale = 10, urg = 5 (h) Scale = 10, urg = 20 (i) Scale = 10, urg = 35

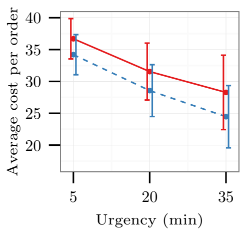
Fig. 3: Comparison with mean relative cost versus dynamism for all levels of scale and urgency. The error bars indicate one standard deviation around the mean.

6 Conclusion

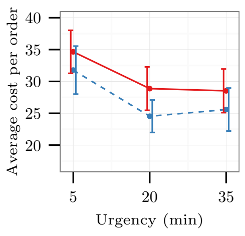
In this paper we present an open source dataset generator and benchmark dataset instance of dynamic PDPTW with support for varying levels of dynamism, urgency and scale. We demonstrate how to use the benchmark instance to compare a decentralized MAS with a centralized algorithm. Although both algorithms are too basic to generalize upon the results, this demonstration can form a baseline to which future work can compare to. Using the work presented in this paper, other researchers in the MAS and OR domains are empowered to conduct thorough and systematic evaluations of their work. In our next paper we plan to reap the benefits of this work by extending the comparison demonstration with a state of the art centralized algorithm and an advanced MAS.

Acknowledgements

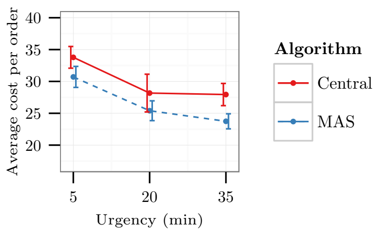
This research is partially funded by the Research Fund KU Leuven.



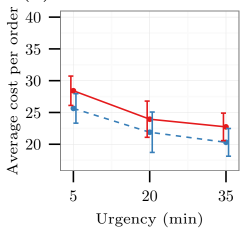
(a) Scale = 1, dyn = 20



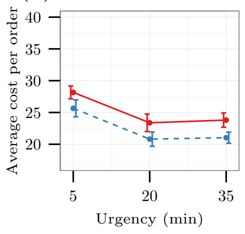
(b) Scale = 1, dyn = 50



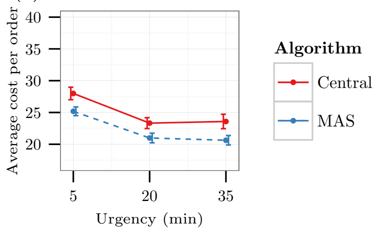
(c) Scale = 1, dyn = 80



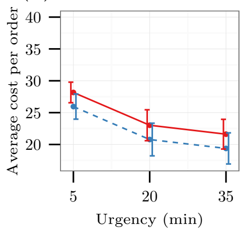
(d) Scale = 5, dyn = 20



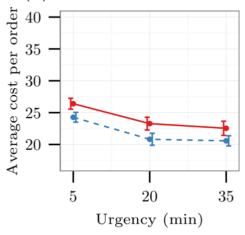
(e) Scale = 5, dyn = 50



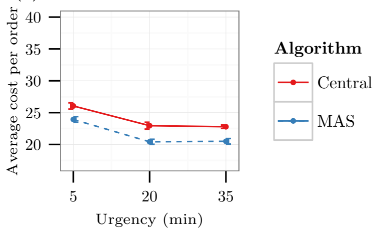
(f) Scale = 5, dyn = 80



(g) Scale = 10, dyn = 20



(h) Scale = 10, dyn = 50

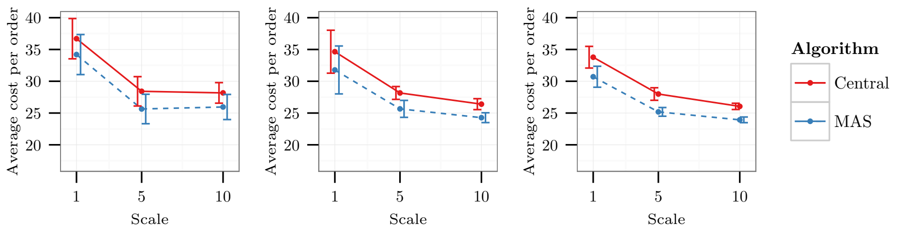


(i) Scale = 10, dyn = 80

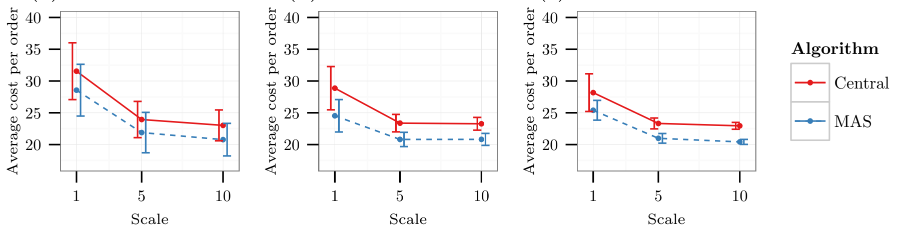
Fig. 4: Comparison with mean relative cost versus urgency for all levels of scale and dynamism. The error bars indicate one standard deviation around the mean.

References

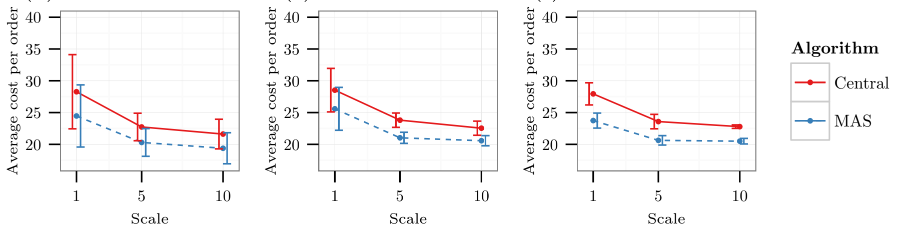
1. Berbeglia, G., Cordeau, J.F., Laporte, G.: Dynamic pickup and delivery problems. European Journal of Operational Research 202(1), 8–15 (2010), doi:10.1016/j.ejor.2009.04.024
2. Coslovich, L., Pesenti, R., Ukovich, W.: A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. European Journal of Operational Research 175(3), 1605 – 1615 (2006), doi:10.1016/j.ejor.2005.02.038
3. Fischer, K., Müller, J.P., Pischel, M.: A model for cooperative transportation scheduling. In: Proc. of the 1st Int. Conf. on Multiagent Systems (ICMAS'95). pp. 109–116. San Francisco (1995)
4. Gendreau, M., Guertin, F., Potvin, J.Y., Séguin, R.: Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. Transportation Research Part C: Emerging Technologies 14(3), 157–174 (2006), doi:10.1016/j.trc.2006.03.002
5. Gendreau, M., Potvin, J.Y.: Dynamic vehicle routing and dispatching. In: Crainic, T., Laporte, G. (eds.) Fleet Management and Logistics, pp. 115–126. Centre for Research on Transportation, Springer US (1998), doi:10.1007/978-1-4615-5755-5_5



(a) urg = 5, dyn = 20 (b) urg = 5, dyn = 50 (c) urg = 5, dyn = 80



(d) urg = 20, dyn = 20 (e) urg = 20, dyn = 50 (f) urg = 20, dyn = 80



(g) urg = 35, dyn = 20 (h) urg = 35, dyn = 50 (i) urg = 35, dyn = 80

Fig. 5: Comparison with mean relative cost versus scale for all levels of urgency and dynamism. The error bars indicate one standard deviation around the mean.

6. Glaschenko, A., Ivaschenko, A., Rzevski, G., Skobelev, P.: Multi-agent real time scheduling system for taxi companies. In: Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS). pp. 29–36 (2009)
7. Gunther, N.J.: Guerrilla Capacity Planning: A Tactical Approach to Planning for Highly Scalable Applications and Services. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006), doi:10.1007/978-3-540-31010-5
8. Ince, D.C., Hatton, L., Graham-Cumming, J.: The case for open computer programs. Nature 482(7386), 485–8 (2012), doi:10.1038/nature10836
9. van Lon, R.R.S.: Code and results, PRIMA 2015 (Aug 2015), doi:10.5281/zenodo.27365
10. van Lon, R.R.S.: Dynamism, urgency and scale dataset (Aug 2015), doi:10.5281/zenodo.27364
11. van Lon, R.R.S.: PDPTW dataset generator: v1.0.0 (Aug 2015), doi:10.5281/zenodo.27362
12. van Lon, R.R.S.: RinLog: v2.0.0 (Aug 2015), doi:10.5281/zenodo.27361
13. van Lon, R.R.S.: RinSim: v4.0.0 (Aug 2015), doi:10.5281/zenodo.27360

14. van Lon, R.R.S., Ferrante, E., Turgut, A.E., Wenseleers, T., Vanden Berghe, G., Holvoet, T.: Measures for dynamism and urgency in logistics. In: CW Reports, vol. CW686. Department of Computer Science, KU Leuven (August 2015)
15. van Lon, R.R.S., Holvoet, T.: RinSim: A simulator for collective adaptive systems in transportation and logistics. In: Proceedings of the 6th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2012). pp. 231–232. Lyon, France (2012), doi:10.1109/SASO.2012.41
16. van Lon, R.R.S., Holvoet, T.: Evolved multi-agent systems and thorough evaluation are necessary for scalable logistics. In: 2013 IEEE Workshop on Computational Intelligence In Production And Logistics Systems (CIPLS), pp. 48–53 (2013), doi:10.1109/CIPLS.2013.6595199
17. Madsen, O.B.G., Ravn, H.F., Rygaard, J.M.: A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. Annals of Operations Research 60(1), 193–208 (1995), doi:10.1007/BF02031946
18. Máhr, T., Srour, J.F., de Weerdt, M., Zuidwijk, R.: Agent performance in vehicle routing when the only thing certain is uncertainty. In: Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS). Estorial, Portugal (2008)
19. Máhr, T., Srour, J.F., de Weerdt, M., Zuidwijk, R.: Can agents measure up? A comparative study of an agent-based and on-line optimization approach for a drayage problem with uncertainty. Transportation Research: Part C 18(1), 99–119 (2010), doi:10.1016/j.trc.2009.04.018
20. Mes, M., van der Heijden, M., van Harten, A.: Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems. European Journal of Operational Research 181(1), 59–75 (2007), doi:10.1016/j.ejor.2006.02.051
21. Mes, M., van der Heijden, M., Schuur, P.: Look-ahead strategies for dynamic pickup and delivery problems. OR Spectrum 32(2), 395–421 (2010), doi:10.1007/s00291-008-0146-3
22. Mitrović-Minić, S., Krishnamurti, R., Laporte, G.: Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. Transportation Research Part B - Methodological 38(8), 669–685 (2004), doi:10.1016/j.trb.2003.09.001
23. Parragh, S.N., Doerner, K.F., Hartl, R.F.: A survey on pickup and delivery problems. Part II: Transportation between pickup and delivery locations. 58(2), 81–117 (2008)
24. Pillac, V., Gendreau, M., Gueret, C., Medaglia, A.L.: A review of dynamic vehicle routing problems. European Journal of Operational Research 225(1), 1–11 (2013), doi:10.1016/j.ejor.2012.08.015
25. Potvin, J., Dufour, G., Rousseau, J.: Learning Vehicle Dispatching with Linear-Programming Models. Computers & Operations Research 20(4), 371–380 (1993), doi:10.1016/0305-0548(93)90081-S
26. Psaraftis, H.: Dynamic vehicle routing: Status and prospects. Annals of Operations Research 61, 143–164 (1995), doi:10.1007/BF02098286
27. Savelsbergh, M.W.P., Sol, M.: The General Pickup and Delivery Problem. Transportation Science 29(1), 17–29 (1995), doi:10.1287/trsc.29.1.17
28. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35(2), 254–265 (1987), doi:10.1287/opre.35.2.254
29. Weyns, D., Boucké, N., Holvoet, T.: Gradient field-based task assignment in an agv transportation system. In: Proc. of 5th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS). pp. 842–849 (2006), doi:10.1145/1160633.1160785