A buffering-strategy-based re-optimization approach to dynamic pickup and delivery problems with time windows

Farzaneh Karami · Wim Vancroonenburg · Greet Vanden Berghe

Abstract It is widely believed that the state-of-the-art in the dynamic pickup and delivery problem with time windows (PDPTW) remains far from reaching maturity, with there being no unanimous agreement with regard to the methods and tools required for obtaining high-quality solutions within a reasonable amount of time. This study proposes a buffering-strategy-based re-optimization approach for dynamic PDPTWs, applied in a computational study on instances with different dynamism degrees and urgency levels. This optimization approach utilizes two heuristics by way of implementing a certain scheduling algorithm: first a cheapest insertion procedure is employed, before subsequently running a local search algorithm. The proposed method's results demonstrate how its performance is impacted by urgency levels, the degree of dynamism associated with request arrivals and the re-optimization frequency. These findings indicate that increases in dynamism reduce costs, whereas increases in urgency result in higher costs. Furthermore, the relative gap to the best found Mixed Integer Linear Programming model solution for which all request information is known in advance is only slightly affected when changing the dynamism degree and urgency level.

Keywords Dynamic pickup and delivery problems · Heuristics · Dynamism · Urgency

1 Introduction

There is a growing consensus in academic literature that many operational level optimization problems such as those found in timetabling and scheduling applications, cannot be dealt with purely from an isolated, static perspective. Planning decisions are made in the context of earlier-made schedules that still impact the current decision problem. New information may be revealed *dynamically* and thus

F. Karami[1] (corresponding author), W. Vancroonenburg[12], G. Vanden Berghe[1]
[1] KU Leuven, Department of Computer Science, CODeS & imec, Gebroeders De Smetstraat 1, 9000 Gent, Belgium
Tel.: +32 9 265 87 04
E-mail: {farzaneh.karami, wim.vancroonenburg, greet.vandenberghe}@cs.kuleuven.be
[2] Postdoctoral research fellow at Flanders Research Foundation - FWO

prompt re-optimization. For example, in the second International Nurse Rostering competition [1] the aspect of *history* was handled by introducing multi-stage nurse rostering, where decisions need to be made on a weekly basis while accounting for historic information from previous weeks on employees' working schedules and performed shifts. Studies on on-line bus prioritization [3] and dynamic patient-to-room assignment [13] further exemplify timetabling applications with the need to address information that is revealed dynamically. Another important area where such aspects cannot be ignored is *logistics*.

Logistics represents a backbone service in several application domains' activities such as in hospitals, distribution companies and storage warehouses. Logistics in these areas requires the continuous picking-up of items from one location and their corresponding delivery to another. Moreover, requests often require handling before a certain time and thus two time windows can be associated with each request: a pickup time window and a delivery time window. From a modeling perspective, this may be interpreted as the scheduling and assignment of tasks to a fleet of vehicles while minimizing tardiness and any other relevant objectives.

In many real-life applications, situations occur in which request information regarding both the number of requests and their respective locations and time windows is unavailable in advance. Due to the dynamic and uncertain nature of request arrivals, logistics management falls within the category of *dynamic* pickup and delivery problems with time windows (PDPTWs). Although research regarding the dynamic PDPTW optimization has grown over the last three decades, unlike the static PDPTW, state-of-the-art modeling, optimization and solution evaluation methods are far from reaching high-quality standards [10].

In the dynamic PDPTW, optimization decisions are made using only the data available at that particular moment in time and are therefore likely to deteriorate in quality as new information becomes available. Unserviced requests may be either rerouted or rescheduled in their current route or relocated to other vehicles by solving sequences of PDPTWs (re-optimization). The time intervals associated with re-optimization distinguishes *instant* from *periodic* approaches. Instant re-optimization is performed upon each request's arrival [2, 14], whereas periodic re-optimization postpones its processing until a certain criterion is met.

Dynamic PDPTWs optimization methods must be capable of producing high-quality solutions under time limitations. Possible dynamic PDPTW optimization methods are those which optimally solve the original problem (exact approaches) and those which address it heuristically with the goal of producing high-quality solutions in limited time. While scalability remains a challenge for exact approaches, local search-based heuristics provide a promising set of features: flexibility, adaptivity and speed, albeit with no guarantee of optimality.

Vancroonenburg et al. [14] contribute methods through which hospitals can increase operational efficiency by dynamically allocating transportation tasks to porters. The proposed solution method integrates different hospital logistics processes and helps to reduce waiting times for patients. In their approach, instant re-optimization is made possible by assuming fast communication technology and

by applying a two phase heuristic-based scheduling algorithm that is sufficiently fast. However, the approach actually ignores the computation time which may be troublesome when urgent requests arrive during computations. Moreover, another focus was on the dynamism of request arrivals regardless of their urgency. Both the explicit consideration of computation times (which are never instantaneous) and requests' urgency levels in the re-optimization were the motivating factors for the present study.

Two sets of strategies were proposed for improving the periodic re-optimization methods' performance: *waiting* and *buffering*. Mitrović-Minić et al. [9] proposed two different waiting strategy types: *drive-first* and *wait-first* which require vehicles to either drive immediately or wait as long as possible before driving to their next destination, respectively. Since this approach is concerned with minimizing travel time, it does not accommodate request tardiness. In contrast to waiting strategies, buffering strategies postpone the least urgent requests until the latest possible time. Pureza and Laporte [11] introduced a new optimization approach for the dynamic PDPTW which involves random travel times and is based on a combination of both waiting and buffering strategies. While this approach facilitates the insertion of future requests while minimizing the travel time and number of lost requests, request tardiness remains unaddressed.

Appropriate measures and metrics are crucial when assessing the proposed dynamic optimization algorithm's performance. While it is sufficient to only report the best solution achieved in a static context, in a dynamic context this is not enough. The primary reason being not only that solution costs at each step (or at the end of planning horizon) are crucial but also the importance of numerous other aspects. For example, how well do the methods accommodate continuous modifications? Do frequent request arrivals affect optimization accuracy? Cruz et al. [4] argue there is no unanimously agreed upon criterion for dynamic optimization algorithm evaluation. The measures proposed thus far range from being entirely problem-specific to near-general in nature. A valid performance measurement approach must be general enough to have the ability to evaluate any dynamic optimization algorithm regardless of its exclusive characteristics. However, it should also be capable of providing insight concerning the overall behavior of an algorithm and its solution cost.

This paper introduces an efficient optimization method which seeks to organize sets of dynamic requests, allocate them to available vehicles, and minimize the requests' service tardiness while respecting several constraints. This optimization method utilizes two heuristics by way of implementing a certain scheduling algorithm: first a cheapest insertion procedure (denoted **CI**) is employed, before subsequently running a local search algorithm (denoted **LS**). For evaluating the behavior of the proposed algorithm, two performance metrics, *local gap* and *global dynamic gap* are employed. These metrics represent the relative gap of the proposed method by comparing its results against those of a Mixed Integer Linear Programming (MILP) model for which all information is known in advance.

2 Problem statement

The primary elements and attributes associated with the dynamic PDPTW are:

Location The locations of pickup and delivery requests and their connections are represented by an undirected *complete* graph $G(V, A, C)$. V is the set of vertices corresponding to all locations. A denotes the set of edges connecting the vertices. Finally, $C : A \to \mathbb{R}$ denotes a travel time function, mapping an edge to its travel time.

Task Each individual pickup and delivery corresponds to a single task.

Service time The time required to perform task i.

Task time window The time window of each task i is expressed using two values: the time window's beginning and end which are denoted by st_i and dt_i, respectively.

Request A request corresponds to a transportation demand and is denoted by $r = (i, j) \in R$, with R representing the set of all requests and $i,\ j \in T$ (T: the set of all tasks). Each request consists of a pickup task i at location $v_i \in V$ and a delivery task j at location $v_j \in V$. A semi-soft time window $[st_i, dt_i)$ is associated with pickup task i thereby implying that it may not be scheduled before st_i, however it may be scheduled after dt_i. Another time window $[st_j, dt_j)$ is specified for delivery task j.

Arrival time This refers to the time at which a request arrives. Each request r's arrival time is denoted by a_r. It is assumed in this paper that both the pickup and the delivery tasks associated with a request are known once the request arrives.

Vehicle Vehicles execute tasks. Each vehicle k has an availability time window $[st_k, dt_k)$ during which tasks may be assigned to it. All vehicles begin and end their working day from a certain location called *depot* $\in V$.

Urgency window (UW) Each request $r \in R$ is associated with a parameter indicating its urgency level, denoted UW, which consists of the time period from its arrival time until the end of its pickup time window. This definition implies the smaller UW values correspond to more urgent requests.

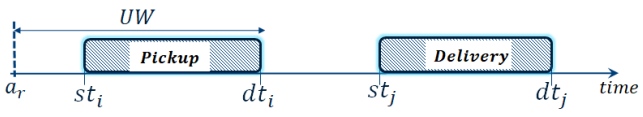

Figure 1: Pickup and delivery time windows and UW

Events In the dynamic PDPTW, requests arrive dynamically and each vehicle's schedule is therefore continuously updated. In this paper, the dynamic PDPTW is modeled and solved in a discrete event-based framework which involves three event types: *request arrival*, *vehicle dispatch* and *vehicle service end*. *Request arrival*: for each request arrival, a corresponding event is generated; *Vehicle dispatch*: when a vehicle departs towards a pickup/delivery location, an event is generated to hold the data of that vehicle's current state and its destination; *Vehicle service end*: when a vehicle finishes a task and is available once again for performing other tasks, an event is generated to hold the data of the vehicle's current state and the task it has just finished.

Scenario In the dynamic PDPTW, a scenario is defined as a triple (τ, ε, P) [5]. Here, τ is the planning horizon length and P is a fleet of vehicles. Meanwhile, ε consists of all request arrival events which are declared between 0 and τ.

Dynamism and urgency The concept of dynamism in the context of the dynamic PDPTW was first introduced by Lund et al. [8] as the ratio of on-line to off-line requests. This ratio was referred to as the degree of dynamism. Later, this definition was refined by Larsen et al. [6] and different classes of dynamic applications were distinguished as either weakly, moderately or strongly dynamic. Recently van Lon et al. [7] have demonstrated that one drawback of this definition is that by only considering the number of request arrivals and neglecting their arrival times, it is possible to overlook the different behaviours associated with these classifications. In their proposed definition, the concept of urgency is considered independent of dynamism and an entirely separate characteristic. In addition, their definition is independent of the entire planning horizon length (τ), thereby making it possible to compare scenarios of different durations. In order to incorporate both characteristics into a single definition, new formal definitions for dynamism and urgency were proposed by van Lon et al. [7]. In these definitions, dynamism relates to the continuity of request arrivals, whereas urgency non-intuitively concerns the maximum time available to handle a request.

Tardiness Request time windows are considered semi-soft: a task may not be scheduled before the start of its corresponding time window, however it may be scheduled after the end of the time window. For each task i, the difference between its completion time Ct_i and its time window end is called tardiness and is calculated as follows: $l_i = Max(0, Ct_i - dt_i)$.

Vehicle overtime Since all requests must be scheduled, vehicles may be assigned requests which lie outside of their availability windows. Vehicle overtime is the completion time of the last request performed by the vehicle minus the end of its availability time window, or 0 if negative.

Travel time For each vehicle, travel time corresponds to the total time it takes to travel between pickup/delivery locations, in addition to its travel time to/from the depot.

The objective function seeks to minimize the sum of three components, total

tardiness, vehicle overtime and total travel time, which are weighted using different coefficients depending on the application/use case.

The fleet of vehicles is assumed homogeneous, all vehicles drive at a constant speed and their cargo default capacity is infinite (e.g. courier service). Each location can be reached directly from any other location via its shortest path calculated on graph $G(V, A, C)$. Each scenario ends when all of its pickups and deliveries have been handled.

A basic yet common method for solving a dynamic PDPTW is by re-optimization. This method considers the dynamic PDPTW MILP formulation, proposed by Savelsbergh and Sol [12], and applies it to the latest updated requests and vehicle data (taking into account what currently is being executed or has already been executed). Thus the MILP model is employed and adapted so as to accommodate the dynamic nature of the PDPTW. For instance, the last-known location of each vehicle is introduced as its depot for the next re-optimization. Additionally, a new time window constraint is introduced to update the availability time window of all currently-dispatched vehicles.

3 Solution approach

In the dynamic PDPTW request arrivals modify the input element of the next re-optimization step. To accommodate this modification, a new problem must be solved. Therefore, any dynamic optimization approach for the dynamic PDPTW consists of solving a sequence of static sub-problems as re-optimization steps. At each step, all unserviced requests are either rerouted or rescheduled. The amount of time between two consecutive re-optimization steps is called a step interval.

In the present study the length of each re-optimization step is referred to as its execution time and is denoted by ET. This value limits the amount of computational time available for an algorithm to generate a solution, taking into account that the earliest decision cannot begin until ET time units after the current step's start time. At each step t, every request is in one of the following four states: (1) **unprocessed** (UR_t): the request has arrived but has not yet been scheduled, (2) **currently processing** (CP_t): the request is scheduled but not-yet-executed, (3) **currently executing** (CE_t): the request is currently scheduled and in progress and (4) **finished** (FR_t): the request has already been scheduled and its corresponding pickup and delivery tasks have been serviced, so it no longer requires processing. At time t (the beginning of a step) the set of all requests eligible for (re)scheduling is $UR_t \cup CP_t$.

3.1 Components

A visual representation of the buffering-strategy-based dynamic optimization algorithm and its components is illustrated in Figure 2. The algorithm is executed regularly at step intervals $t_i^{exec} = i \times ET$ for $i = 0, 1, 2, \ldots, \frac{\tau}{ET}$. At each step the set of eligible requests is constructed as input to a scheduling algorithm. Inputs

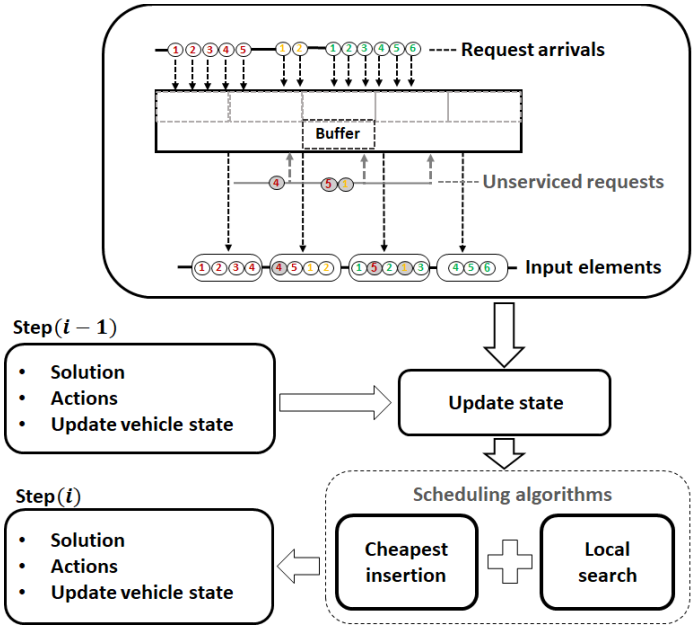

Figure 2: A visual representation of the buffering-strategy-based dynamic optimization approach.

only consist of the requests known and unserviced at that moment in time. In parallel to the scheduling algorithm's execution, any new request arrivals are buffered to be used as input in the next re-optimization step.

The *scheduling algorithms* are used for assigning vehicles to requests and sequencing them. The services performed by vehicles at each step are called *actions*.

3.2 The significant role of ET

Since parameter ET affects the number of requests taken as input at each re-optimization step, a large ET value may postpone some eligible requests for a significant amount of time, which may consequently contribute towards additional tardiness. Thus, ET should be set in accordance with the urgency of request arrivals.

Assume the urgency window of the most urgent request is known (denoted by UW_{min}) and the value of ET is set to a value less than or equal to it ($ET \leq UW_{min}$). As a result, there are no requests with UW smaller than ET in the requests list. Ideally ET should be set equal to UW_{min} because any smaller value represents a waste of computational resources. Since in most dynamic applications, UW_{min} is unknown in advance, some investigation is required to isolate an appropriate value for ET. Figure 3 illustrates the pickup time window's relationship with ET and UW.

According to van Lon et al. [7] dynamism and urgency are two separate con-

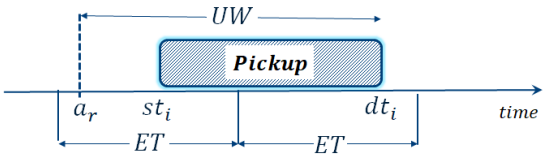

Figure 3: A visualization of ET and UW

cepts, each affecting the problem and optimization approach in a different way. Therefore, in scenarios of the same scale, the value of ET should be larger in those with less urgent request arrivals but similar dynamicity. The scale is a combination of the number of vehicles, number of requests and area size. The ET value also depends on the travel time to the eligible requests with UW_{min} or in general to the scale of the problem. For a fixed number of vehicles and requests, by increasing the size of area, the average travel time of all requests will also increase. Thus, a full data investigation is required to isolate an appropriate value for ET.

3.3 **Scheduling algorithms**

The Cheapest Insertion and Local Search *scheduling algorithms* are taken and adapted from [14] and are detailed next:

Cheapest insertion The CI policy inserts eligible requests at each step based on the current state, with the goal of minimally increasing the objective function value. Certain constraints such as time windows must be respected to guarantee feasibility. After deciding on a feasible insertion position in a vehicle's list of scheduled tasks, the start and end times of all tasks from that list are recalculated. This is performed by iterating over all the tasks contained in the list and updating their start times[1] based on the maximum sum of the current decision time, the travel time to the pickup/delivery location and start of the corresponding time window st_i.

Local search The local search policy represents an iterative application of the cheapest insertion policy. A set of eligible requests is inserted into the solution at each step interval using the aforementioned CI method, before the LS algorithm iteratively applies a simple local-search-based procedure of ejection, move and reinsertion. At each local-search iteration, a request is randomly selected and ejected from its corresponding vehicle's schedule before the CI procedure subsequently inserts it into a (potentially different) vehicle's schedule. The start and end times of the scheduled tasks associated with these vehicles are updated. A new solution is accepted only if its cost is lower than or equal to the best known cost from previous solutions. This iterative search continues until a predefined number of

[1] This is the start time of the pickup/delivery service, not the dispatch time to the location.

maximum non-improving iterations is reached.

This paper assumes step intervals are long enough to enable the search process to complete. The following stopping criterion must therefore be included so as to ensure this: iterations must stop either before the next step or after a maximum number of non-improving iterations is achieved. Additionally, the earliest scheduled departure time of a vehicle must occur after $ET + t$, where t is the last step's start time. Earlier scheduled departure times would imply the vehicle knows it has to depart to a task location before the algorithm has finished its calculations, which is clearly infeasible.

3.4 **Dynamic optimization algorithm**

Ultimately, the proposed dynamic optimization algorithm functions according to the following steps:

1. The entire planning horizon is subdivided into a number of step intervals of length ET.
2. Based on recent events, previous decisions and actions, a request queue consisting of all eligible input elements is constructed at the beginning of each step $t := i \times ET$ for $i = 0, 1, 2, \ldots, \frac{\tau}{ET}$.
3. A set of rules is applied which stipulate the conditions under which the scheduling algorithm must make decisions.
4. The scheduling algorithm is executed on the constructed request queue.
5. The scheduling algorithm's outcomes and actions are stored at each step.

4 Computational study

A computational study is performed to evaluate the dynamic optimization approach and LS scheduling algorithm. This study focuses on comparing the dynamic optimization algorithm's solution and total actions against the best static solution found by the MILP model, for which all information is known in advance.

4.1 **Performance evaluation criteria**

This section proposes two performance metrics for evaluating the behavior of the proposed algorithm: local gap and global dynamic gap.

Local gap At each step, this gap compares the relative optimality gap of the LS scheduling algorithm's local solution against the optimum static solution obtained by the MILP model for the same input. A dynamic algorithm is considered superior to another if it generates solutions closer to optimality throughout the entire optimization process. Local gaps (LGs) are calculated for each step i by $LG_i = 100 \times (\frac{Alg(\theta_i) - OPT(\theta_i)}{Alg(\theta_i)})$, where $OPT(\theta_i)$ is the cost of either the optimum or the best solution found by the MILP model for each request queue θ_i, with $Alg(\theta_i)$ denoting the cost of step i using the LS.

Global dynamic gap The overall dynamic optimization's solution is not necessarily optimal, even though each individual re-optimization may have been solved to optimality. Since the local gap only allows to draw conclusions regarding the scheduling algorithms' ability to find good quality solutions to each re-optimization problem, the global dynamic gap is considered. This compares the global dynamic solution cost, the cost obtained by the dynamic approach over the entire scenario by repeatedly applying the scheduling algorithm, against the best solution obtained by MILP model over the entire planning horizon. To this end, the global dynamic gap (GDG) is calculated for the entire planning horizon by $GDG = 100 \times (\frac{Alg(\theta') - OPT(\theta')}{Alg(\theta')})$, where $OPT(\theta')$ denotes the best static solution found by the MILP model for all requests θ', with $Alg(\theta')$ denoting the cost of the solution obtained by applying LS throughout the scenario.

In the context of dynamic optimization, an algorithm performs well if request characteristics have a minimal impact upon its GDG. To study the impact of step length on objective value and relative optimality gaps, a set of hypotheses is considered. These hypotheses concern the extent to which the global dynamic cost and its relative optimality gap are affected by the dynamism and urgency of request arrivals. For example, by decreasing dynamism and increasing urgency, the cost will increase and the GDG will slightly increase. On the contrary, by increasing dynamism and decreasing urgency, the cost will decrease and the GDG will slightly decrease.

4.2 **Datasets**

The present computational study employs a dataset created by the generator introduced by van Lon et al. [7]. This dataset consists of different configurations of dynamism degrees and urgency levels for request arrivals. In doing so, it enables evaluating the proposed approach in various distinct situations.

The generator was employed to generate scenarios with five different degrees of dynamism 20%, 40%, 50%, 80% and 90% and five different levels of urgency 5, 15, 25, 35 and 45 minutes. Five instances were produced for each possible setting of these dynamism degrees and urgency levels resulting in 125 scenarios in total.

For example, to generate instances with different degrees of dynamism, consider $\triangle := \{\delta_0, \delta_1, \ldots, \delta_{|\varepsilon|-2}\} = \{a_{rj} - a_{ri} | j = i + 1 \wedge \forall r_i, r_j \in \varepsilon\}$, which represents the sequence of inter-arrival times for requests and $| \triangle | := |\varepsilon| - 1$. The perfect inter-arrival time required for 100 percent dynamism is $\frac{\tau}{|\varepsilon|}$.

One possible example sequence is $\triangle_a = \{0.1, 2, 0.1, 2, 0.1, 2, 0.1, 2, 0.1\}$ which includes five small bursts with intervals 0.1 and 2 units (Figure 4-left). By changing this order, another possible sequence for request inter-arrival time is $\triangle_b = \{0.1, 0.1, 0.1, 0.1, 0.1, 2, 2, 2, 2\}$ which has one big burst of request arrivals at the beginning and four individual request arrivals (Figure 4-right). Based on van Lon's definition, dynamism is measured by $1 - \frac{\sum_{i=0}^{|\triangle|} \sigma_i}{\sum_{i=0}^{|\triangle|} \acute{\sigma}_i}$, for which the numerator is the sum of all deviations of an inter-arrival time relative to the 100 percent case (the

deviation of an entire scenario) and for which the denominator is the maximum deviation for a scenario. Thus, Figure 4-left represents 50% dynamism and Figure 4-right depicts a degree of dynamism equal to 27.5%.

Figure 4: Ten request arrival events which exhibit 5 inter-arrival times of 0.1 and 4 inter-arrival times of 2 [7].

Regarding the study by van Lon et al. [7] the pickup/delivery task time windows have been randomly generated while respecting the constraints on different request urgency levels (UW is expressed in minutes) and the announce times.

Simulations were run for a total of 10 vehicles with an assumed 50 km/h velocity. The service time of each task is 5 *minutes* and the entire planning horizon's length is one day.

4.3 **Computational results**

Computational experiments were performed on a computer with an Intel Xeon, E5-2640, 2.60GHz processor with 16GB of RAM memory running Linux Ubuntu 16.04.3. The MILP formulation was solved by Gurobi 7. The scheduling algorithm's runtime was limited to ET for each step interval. The proposed dynamic optimization approach was applied within a simulation environment, implemented in Java 1.8.

In order to evaluate the scheduling algorithm, its local and global performance were assessed by calculating LG_i at each step as well as the GDG after the final step. To assess the local performance at each step first, the scheduling algorithm was executed on the constructed request queue and the resulting cost in addition to all actions were stored. The MILP model was subsequently executed on the same request queue and its results were also stored. This provided the necessary data for assessing local performance. To assess the global performance, each scenario was solved using the MILP model with all information known in advance. All these simulations were performed using the same ET.

Local gap Figure 5 presents two boxplot graphs. The left boxplot indicates the local solution gap obtained by LS for scenarios with different urgency levels (UW values), wherein each box aggregates local solutions of 15 instances with three different dynamism degrees and the same urgency level. The right boxplot presents the local solution gap for all dynamism degrees, where each box aggregates local solutions of 25 instances with five different urgency levels and the same dynamism degree. Although scenarios may differ considerably from each other, it is possible to aggregate them in Figure 5. From this figure it can be concluded that in high urgency instances (the smaller the UW value, the more urgent the request) better

local solutions on average can be obtained. This is expected because of the small number of requests in the eligible request list, which makes the LS more effective. It also reveals how the maximum of the local gaps decreases when dynamism increases or urgency decreases. Its small increment on non-urgent scenarios may be explained by the increased total number of eligible requests at each step which may demand for more computation time to achieve a tighter gap.

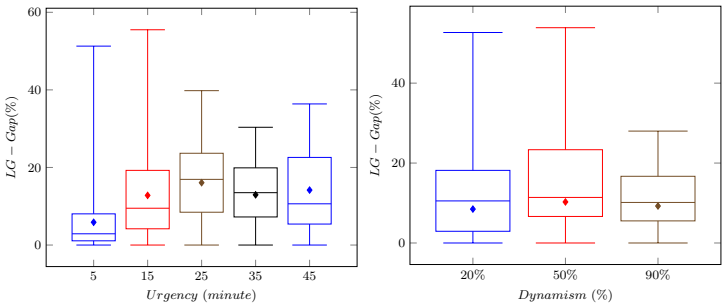

Figure 5: Local gaps (LGs) on scenarios with different levels of urgency (UW values) 5, 15, 25, 35 and 45 minutes and three degrees of dynamism 20%, 50% and 90%. In the left boxplot, each box aggregates LGs for scenarios with three different dynamism degrees and the same urgency level. The right boxplot presents the LGs for scenarios with five different urgency levels and the same dynamism degree. The average gap for each boxplot is represented by '•'.

Global solutions The proposed dynamic optimization algorithm is analyzed based on the experimental results regarding whether or not the proposed optimization method's cost and GDG depend on the degree of dynamism and the urgency level of incoming requests. Figures 6(a)-6(c) and 7(a)-7(c) illustrate the experimental results concerning cost and GDG, respectively.

Figure 6(a) illustrates how for any level of urgency (UW value) the cost will decrease by increasing the degree of dynamism. Also, Figure 6(b) demonstrates how the solution cost will decrease by decreasing the urgency level of request arrivals for any dynamism degree. Figure 6(c) illustrates how instances with low dynamism degrees and high levels of urgency are the most challenging and costly instances for the proposed dynamic algorithm to solve.

Figures 7(a)-7(c) illustrate the primary achievement of the proposed approach. The observation is that GDG is only slightly affected by the dynamism and urgency of request arrivals. Regarding GDG behavior, by increasing the dynamism degree for a low level of urgency, decreases in the GDG continue until a certain degree of dynamism is reached. For high urgency levels, the GDG increases for high degrees of dynamism while staying within the range of 15%-37%. The main reason why this gap slightly increases for very high degrees of dynamism is the

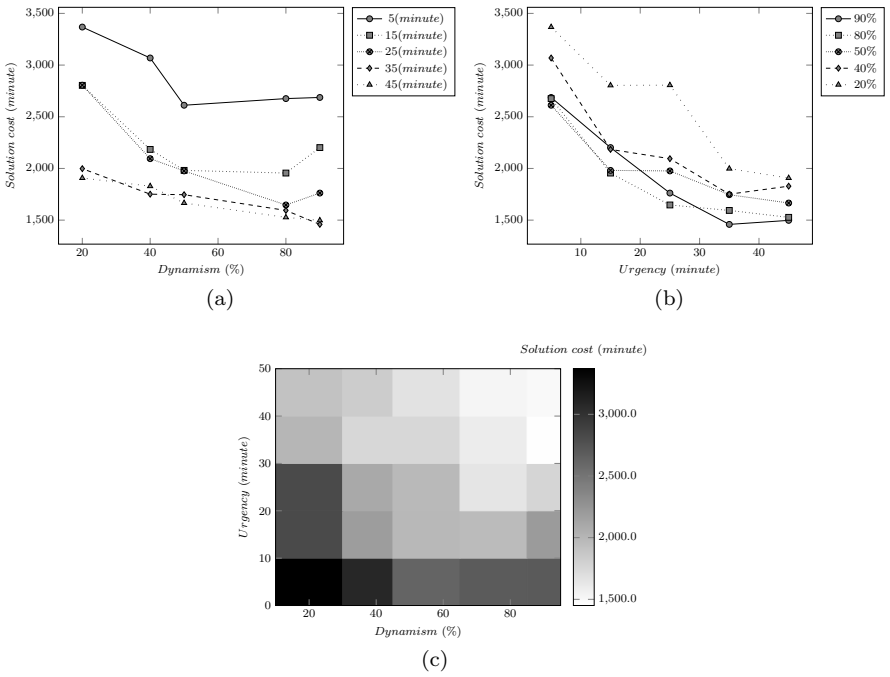

Figure 6: Solution cost using the LS. Figure (a) illustrates that the cost will decrease by increasing the degree of dynamism. Figure (b) demonstrates that the solution cost will decrease by decreasing the UW value. Figure (c) illustrates how often instances of varying degrees of dynamism and urgency levels are the most/least costly.

more significant role data uncertainty plays when increasing the dynamism. Another reason for that is increasing the queue length. A longer queue may require more computation time than the time limit (ET) to achieve the same gap. Thus, request queue length and computation time limit (ET) function as two interrelated parameters primarily responsible for affecting the GDG. If ET is too small then a large gap is likely introduced. On the other hand, if ET is too large then the request queue length may increase which in turn will also result in large gaps.

5 Conclusions and future work

Almost all real-world logistics problems are subject to an information release over time, necessitating repeated decision making over time. Although over the last two decades significant research effort has been allocated to dynamic optimization, there is no agreed groundwork of methods and tools for comprehensive algorithm analysis in dynamic optimization. This paper helps to remedy this deficiency.

The proposed buffering-strategy-based optimization approach for the dynamic PDPTW is designed to take advantage of special problem structures and therefore

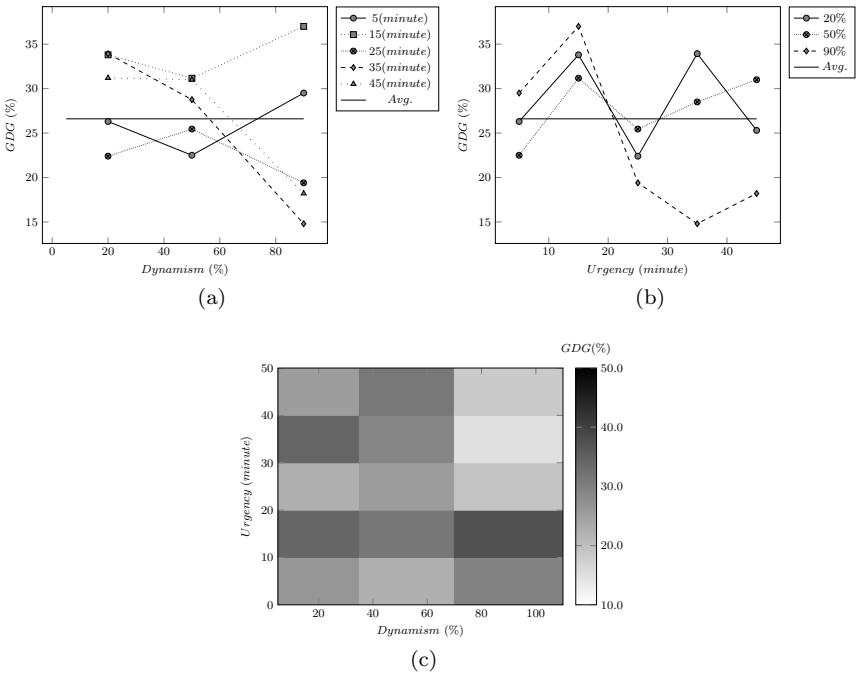

Figure 7: GDGs result using the LS. Figures (a) and (b) demonstrate that GDGs in the scenarios with different urgency levels and dynamism degrees are slightly affected by variations of these two request arrivals characteristics.

better utilize heuristic methods. Particular attention was given to on-line high-quality solution construction, not only at each step (local perspective) but also at the end of the planning horizon (global perspective). To evaluate the proposed approach, the gap between the best found MILP cost and the heuristic-based dynamic optimization cost was studied from both local and global perspectives.

It was also shown how urgency levels, dynamism degrees of request arrivals, and re-optimization frequency all impact cost and the relative gap to the best found MILP solution for which all information is known in advance. Results indicated how any increase in dynamism results in decreased costs, whereas any increment in urgency will have the opposite effect. In addition, it is noteworthy that the relative gap is only slightly affected by any changes regarding the degree of dynamism or the urgency of request arrivals.

Different aspects of the proposed dynamic optimization approach are currently being studied. Future research directions may include further investigating the sensitivity of the cost and relative gap with regard to the objective weights and ET, adapting of existing approaches for large-scale real-life applications and generating an efficient exact approach for calculating their lower bounds. Another

promising direction involves developing a robust optimization approach for the dynamic PDPTW.

Acknowledgements W. Vancroonenburg is a postdoctoral research fellow at Research Foundation Flanders - FWO Vlaanderen. Editorial consultation provided by Luke Connolly (KU Leuven).

References

1. Ceschia, S., Thi Thanh Dang, N., De Causmaecker, P., Haspeslagh, S., Schaerf, A.: Second International Nurse Rostering Competition (INRC-II) — Problem Description and Rules —. ArXiv e-prints (2015)
2. Chang, M.S., Chen, S.R., Hsueh, C.F.: Real-time vehicle routing problem with time windows and simultaneous delivery/pickup demands. Journal of the Eastern Asia Society for Transportation Studies **5**, 2273–2286 (2003)
3. Cogill, R., Mareček, J., Mevissen, M., Rudová: Online problems in timetabling: Bus priority at signalised junctions. In: Proceedings of the 11th international conference on the practice and theory of automated timetabling, Udine, Italy, 23-26 August, pp. 81–94 (2016)
4. Cruz, C., González, J.R., Pelta, D.A.: Optimization in dynamic environments: a survey on problems, methods and measures. Soft Computing **15**(7), 1427–1448 (2011)
5. 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)
6. Larsen, A., Madsen, O., Solomon, M.: Partially dynamic vehicle routing—models and algorithms. Journal of the Operational Research Society **53**(6), 637–646 (2002). DOI 10.1057/palgrave.jors.2601352. URL https://doi.org/10.1057/palgrave.jors.2601352
7. van Lon, R.R., Ferrante, E., Turgut, A.E., Wenseleers, T., Vanden Berghe, G., Holvoet, T.: Measures of dynamism and urgency in logistics. European Journal of Operational Research **253**(3), 614–624 (2016)
8. Lund, K., Madsen, O.B.G., Rygaard, J.M.: Vehicle routing problems with varying degrees of dynamism (tech. rep.). Lyngby, Denmark: IMM, The Department of Mathematical Modelling, Technical University of Denmark (1996)
9. 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)
10. Psaraftis, H.N., Wen, M., Kontovas, C.A.: Dynamic vehicle routing problems: Three decades and counting. Networks **67**(1), 3–31 (2016)
11. Pureza, V., Laporte, G.: Waiting and buffering strategies for the dynamic pickup and delivery problem with time windows. INFOR: Information Systems and Operational Research **46**(3), 165–175 (2008)
12. Savelsbergh, M.W., Sol, M.: The general pickup and delivery problem. Transportation Science **29**(1), 17–29 (1995)
13. Vancroonenburg, W., De Causmaecker, P., Vanden Berghe, G.: Patient-to-room assignment planning in a dynamic context. In: Proceedings of the 9th

International Conference on the Practice and Theory of Automated Timetabling, Son, Norway, 28–31 August, pp. 193–208 (2012)
14. Vancroonenburg, W., Esprit, E., Smet, P., Vanden Berghe, G.: Optimizing internal logistic flows in hospitals by dynamic pick-up and delivery models. In: Proceedings of the 11th international conference on the practice and theory of automated timetabling, Udine, Italy, 23-26 August, pp. 371–383 (2016)