Robustness of periodic reoptimization policies for the dynamic PDPTW

Farzaneh Karami · Wim Vancroonenburg · Greet Vanden Berghe

the date of receipt and acceptance should be inserted later

Abstract In the dynamic pickup and delivery problem with time windows (PDPTW), there exists uncertainty concerning the time windows and locations associated with requests. Periodic reoptimization policies are suitable for dealing with such uncertainty. The key question is to what extent these policies are robust for the dynamic PDPTW. We analyze robustness by taking into account both the reoptimization period and the dynamism degree. We first evaluate the robustness locally after each reoptimization and then evaluate it globally at the end of the scheduling horizon. Results indicate that robustness increases both locally and globally when either the reoptimization period lengthens or the dynamism degree increases, and vice versa. Finally, we conclude that periodic reoptimization policies for the dynamic PDPTW handle uncertainty best when the reoptimization period matches the requests' urgency.

Keywords Dynamic pickup and delivery with time windows · Periodic reoptimization policy · Robustness analysis

1 Introduction

How should one handle the inherent uncertainty present in dynamic problems? This is not a straightforward question to answer given that there is little agreement concerning how exactly one should even quantify performance in dynamic problems, let alone how best to maintain high quality solutions under such conditions. In a problem such as the dynamic pickup and delivery problem with time windows (PDPTW), uncertainty can arise in many forms: request locations, request time windows, vehicle availability and so forth. If an approach for the dynamic PDPTW is able to consistently produce high quality solutions despite these unknowns, then

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

it is referred to as robust.

The more uncertainty a method can handle, the more robust it is said to be. Two possibilities for addressing uncertainty in the dynamic PDPTW are (i) those which reserve unscheduled vehicles to deal with unforeseen request arrivals and (ii) reoptimization-based approaches. Despite their widespread use, reservation-based approaches are inefficient for two reasons. First, surplus vehicles are not profitable during non-peak periods. Second, if none of the reserved vehicles are located close to the location(s) associated with unpredictable requests, then it is a waste of resources to reserve additional vehicles during the peak periods.
As Psaraftis [5] have highlighted, there is usually very little flexibility when it comes to varying fleet size in reoptimization-based approaches. There is usually some timespan before the schedules are executed. Consequently, solving the dynamic PDPTW as a static scheduling problem using robust optimization affords greater flexibility in terms of including additional vehicles in order to meet demand.
Dynamism and urgency [7] are two important parameters associated with a dynamic PDPTW instance. The dynamism degree corresponds to the frequency with which requests arrive. Meanwhile, urgency levels convey how quickly those requests must be serviced. The combination of these two parameters represents an instance's degree of uncertainty. A thorough understanding of the issues stemming from the various forms of uncertainty is helpful for solving the PDPTW. This is especially the case when there is no prior knowledge available concerning request arrivals, their locations and their time windows.
The time interval associated with reoptimization is what determines optimization runtime. One possibility is to react to each and every information update, a policy referred to as reactive reoptimization. Another possibility is to only reoptimize once a set of predefined criteria is met, a policy referred to as periodic reoptimization. Reactive reoptimization is likely to generate huge computational burdens when solving large-scale problems and demands more resources than periodic reoptimization policies to maintain solution quality [3]. On the other hand, the periodic reoptimization policy introduced by Karami et al. [2] takes urgency levels into account and defines a buffering time interval between consecutive calls to the solver. As a result, periodic reoptimization policies are capable of controlling the amount of time available for reoptimization. Thus, they call for a trade-off between solution quality and robustness.
Figure 1 illustrates how utilizing a reactive reoptimization policy differs from using a periodic policy. This example concerns the assignment of two requests to a single vehicle before noon. The two requests are announced during the workday at 9:35 and 9:40 with their pickup and delivery time windows being [9:35-11:00] and [9:40-9:50], respectively. Rejecting requests is not permitted. The vehicle traverses a Manhattan-style grid where each edge requires five minutes of travel time. The objective is to minimize the sum of travel times, driver overtime, and lateness of requests. While the best solution the reactive reoptimization policy can achieve takes 150 minutes, the periodic policy generates a solution of 135 minutes when its reoptimization period is set to 5 minutes, despite the fact it begins servicing at a later point in time. If we increase the reoptimization period from 5 to 20 minutes, the best attainable objective value then becomes 190 minutes. This example simply demonstrates the potential benefit of a periodic reoptimization policy and the cru-

cial role of the reoptimization period. Now, however, one crucial question arises: what reoptimization period will produce high quality solutions that are also robust with respect to unpredictable request arrivals?

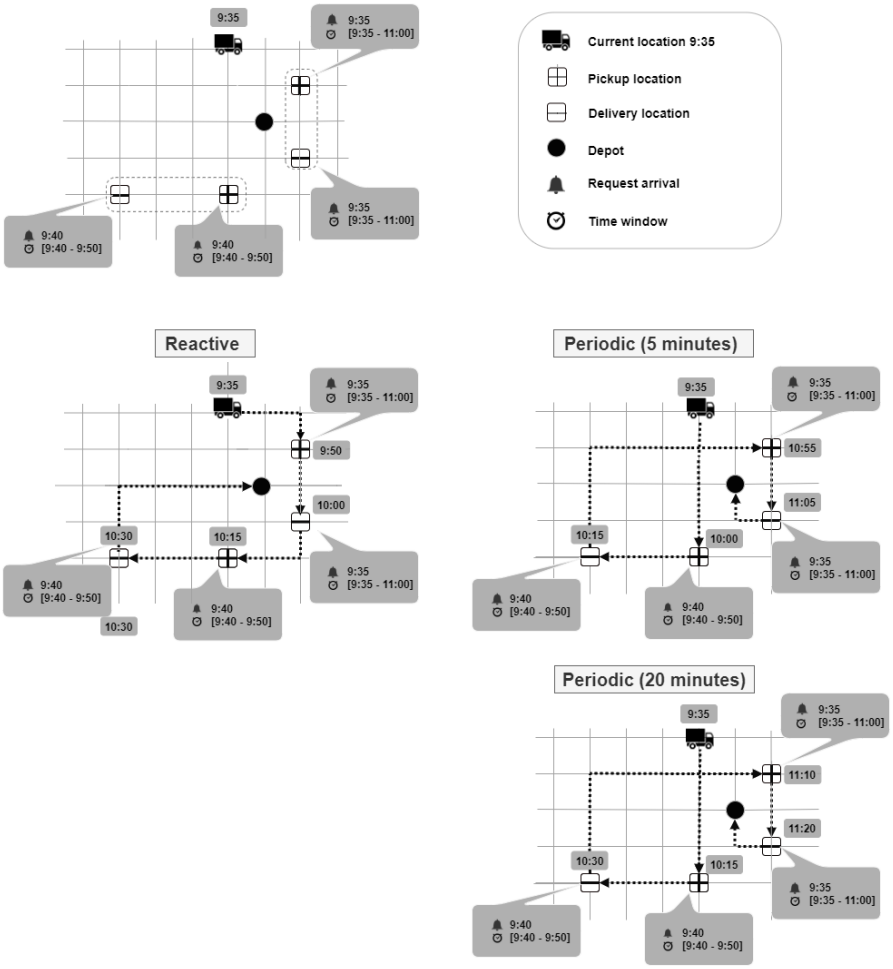


Figure 1: Dynamic PDPTW addressed by one reactive and two periodic reoptimization policies

Two possible metrics for robustness are the solution itself and its quality. In the former, robustness concerns preserving the assignments constituting a solution. Meanwhile, the latter metric only considers solution quality, regardless of the number of changes made to the schedule. Since solutions are frequently updated during a periodic reoptimization policy, the solution quality perspective of robustness

makes most sense. Therefore, for the remainder of this paper, robustness refers to solution quality robustness.
Investigating a periodic reoptimization policy's robustness involves exploring the range of different degrees of uncertainty under which a given dynamic PDPTW performs 'well', 'as intended' or 'as required'. The dynamic PDPTW with *request time windows' uncertainties* has been defined by Srour et al. [6] with the assumption that exact pickup and delivery locations are known while the time windows are uncertain. Morlok and Chang [4], however, consider request location uncertainty for a transportation system.

Nevertheless, the academic literature lacks a means of quantifying robustness when it comes to periodic reoptimization policies for the dynamic PDPTW, when considering uncertainty for both timing (arrivals and time windows) and locations. This research is dedicated to addressing this issue. We analyze the robustness of periodic reoptimization in the dynamic PDPTW with respect to varying reoptimization periods and degrees of dynamism. Our experiments are conducted on generated instances which exhibit a variety of dynamism degrees and urgency levels.

2 Uncertainty metric

2.1 Urgency and dynamism

A dynamic PDPTW instance [1] is a triple (τ, ε, V) where τ denotes the scheduling horizon, V the fleet of vehicles, and where ε consists of all request arrivals. The time at which a request r becomes known is referred to as its arrival time a_r. The continuity of request arrivals corresponds to the PDPTW's degree of dynamism, while urgency is treated as a distinct characteristic and defined as the length of time from a request's arrival until the end of either its pickup or delivery time window. The information regarding both the pickup and delivery tasks' time windows is available upon request arrival. Given that the pickup should be finished first, we define urgency based on the pickup time window.

To generate instances with different degrees of dynamism using the generator provided by van Lon et al. [7] consider $\triangle := \{\delta_0, \delta_1, \ldots, \delta_{|\varepsilon|-2}\} = \{a_{rj} - a_{ri} \mid j = i+1 \ \wedge \ \forall r_i, r_j \in \varepsilon\}$, which represents the sequence of inter-arrival times for requests, where $|\triangle| := |\varepsilon| - 1$.

The inter-arrival time required for uniform distribution, in other words 100 percent dynamism, is $\frac{\tau}{|\varepsilon|}$. Based on the definition provided by van Lon et al. [7], dynamism is measured by

$$Dy = 1 - \frac{\sum_{i=0}^{|\triangle|} \sigma_i}{\sum_{i=0}^{|\triangle|} \hat{\sigma_i}},$$

for which the numerator is the sum of all deviations of inter-arrival times (σ_i) relative to the 100 percent case and where the denominator is the maximum deviation for the scenario.

One possible example sequence of inter-arrival times for requests is $\triangle_a = \{0.1, 1, 0.1, 1, 0.1, 1, 0.1, 1, 0.1\}$, which corresponds to five small bursts with intervals of 0.1 and four of 1 unit, as shown in Figure 2(a). By changing this order, another possible sequence for request inter-arrival times is $\triangle_b = \{1, 1, 0.1, 0.1, 0.1, 0.1, 0.1, 1, 1\}$, which has one large burst of request arrivals in the middle and two individual request arrivals at both the beginning and end, as shown in Figure 2(b). Thus, the degree of dynamism of Figure 2(a) is 55.5%, whereas that of Figure 2(b) is 35.1%.

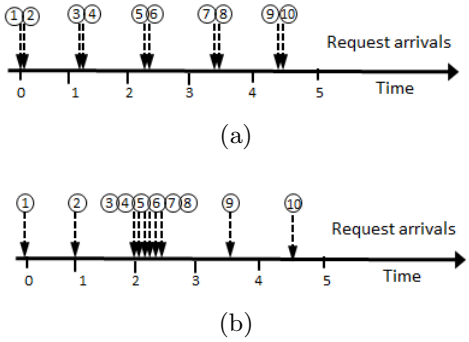


Figure 2: Two possible examples of request arrival events with five inter-arrival times of 0.1 and four inter-arrival times of 1 unit.

In order to conduct our computational study we created a set of instances whose characteristics exhibit a variety of dynamism and urgency combinations. The generator was employed to produce instances with three different degrees of dynamism (20%, 50% and 90%) and five different levels of urgency (5, 15, 25, 35 and 45 minutes). Five instances were produced for each of the 15 possible combinations, resulting in a total of 75 instances.

2.2 An uncertainty metric for periodic reoptimization policy

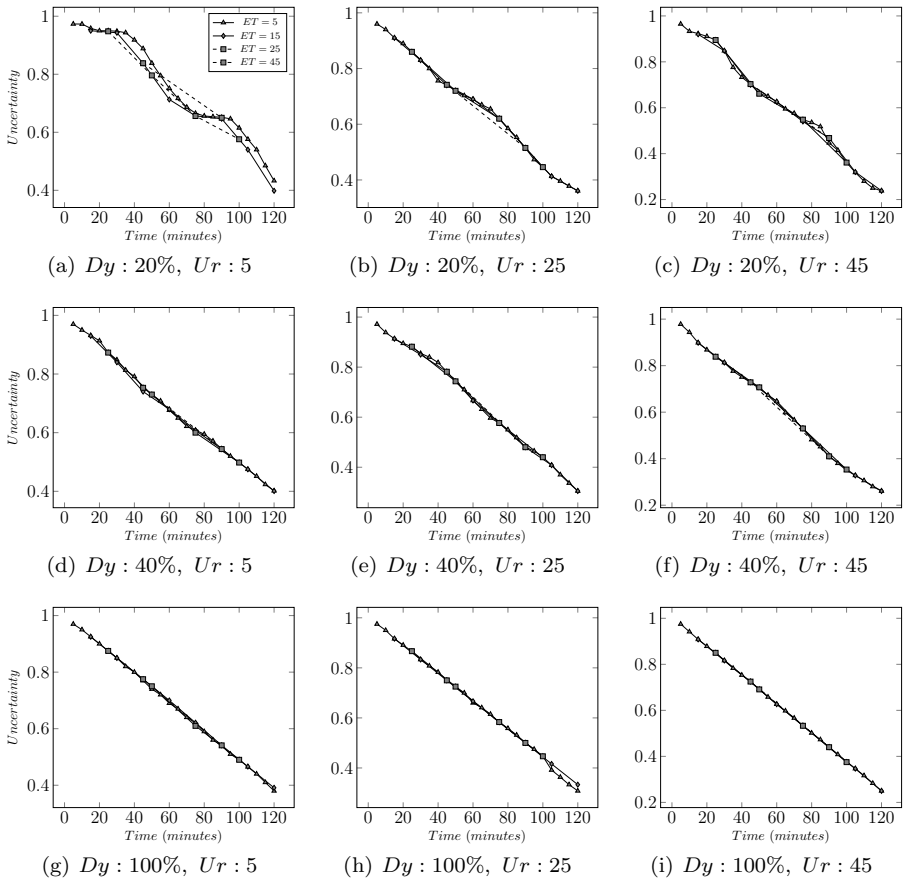
The *degree of certainty* of a periodic reoptimization policy refers to the proportion of observed requests it has integrated into the reoptimization at a particular time t. Let $|\varepsilon|$ be the total number of requests for an instance and n_t the number of requests which have arrived before t. The degree of certainty at time t is then defined as $\frac{n_t}{|\varepsilon|}$.

The length of time between two consecutive optimizations is denoted by ET [2]. Meanwhile, the degree of certainty C_i of reoptimization period $i \geq 1$ is equal to $C_{i-1} + \Gamma_i$, where $C_0 = 0$ and Γ_i represents the additional amount of certainty a periodic reoptimization policy gains during period i, which can be calculated as follows:

$$\Gamma_i = \frac{n_{iET} - n_{(i-1)ET}}{|\varepsilon|}$$

We define the degree of uncertainty at optimization period i as $U_i = 1 - C_i$. We consider instances with a variety of dynamism and urgency configurations in the

first two hours of the scheduling horizon in Figure 3. The graphs show the uncertainty degree determined by a periodic reoptimization policy. The scheduling horizon's length τ is four hours. Dy and Ur denote each instance's dynamism degree and urgency level, respectively. The degree of uncertainty is calculated for various ETs. One can observe nonlinear variations concerning the uncertainty degree of instances with lower degrees of dynamism. However, given that the uncertainty degree increases relative to the reoptimization frequency, one question logically arises: does solution quality deteriorate as uncertainty increases? And if the answer to this question is affirmative, then by how much does it deteriorate?



(a) Dy : 20%, Ur : 5 (b) Dy : 20%, Ur : 25 (c) Dy : 20%, Ur : 45

(d) Dy : 40%, Ur : 5 (e) Dy : 40%, Ur : 25 (f) Dy : 40%, Ur : 45

(g) Dy : 100%, Ur : 5 (h) Dy : 100%, Ur : 25 (i) Dy : 100%, Ur : 45

Figure 3: Uncertainty degrees of periodic reoptimization policy in the first two hours of the scheduling horizon.

3 Periodic reoptimization policies robustness

In this section we analyze the robustness of the periodic reoptimization policy proposed by Karami et al. [2] where additional requests are inserted into the

existing schedule with the objective of minimizing the sum of travel time, vehicle overtime and request lateness. We employed the iterative scheduling algorithm introduced by Vancroonenburg et al. [8]. At each iteration, a random request is eliminated from its associated transporter's route. This request is then reinserted into the schedule using the cheapest insertion procedure. If the resulting solution is better than the current solution, it is accepted. The search stops if a predefined stopping criterion as a maximum number of non-improvements is met. This study will compare the periodic policy's objective values against those of the optimum static solutions obtained by a MILP, which were generated for both the local and the global problems.

Robustness metric A challenge present in periodic reoptimization is how to measure the impact of the degree of uncertainty on an existing schedule, both locally after each optimization and globally at the end of the scheduling horizon. There is no metric available for calculating local or global robustness. However, such a metric is needed to assess the impact of request arrivals on the solution's quality. First, consider the solution quality associated with each of the scheduling horizon's reoptimization periods. Let us then define L_{nr} as the percentage difference between the average solution quality across the entire horizon and the worst quality observed. We also define G_{nr} as the gap between the objective value incurred by the periodic reoptimization policy over the entire scheduling horizon with respect to the optimal objective value over the same period. Therefore, *local and global robustness* can be defined as 100-L_{nr} and 100-G_{nr}, respectively.

Local robustness Figure 4 provides the local robustness values for instances with urgency levels of {5, 15, 25, 35 and 45 minutes} and dynamism degrees of {low-20%, medium-50% and high-90%}. Each node in Figure 4(a) corresponds to the local robustness for instances with five different urgency levels and identical dynamism degrees. Similarly, each node in Figure 4(b) corresponds to the local robustness for instances with three different dynamism degrees and identical urgency levels. Note that progressing along the x-axis involves the level of urgency decreasing.

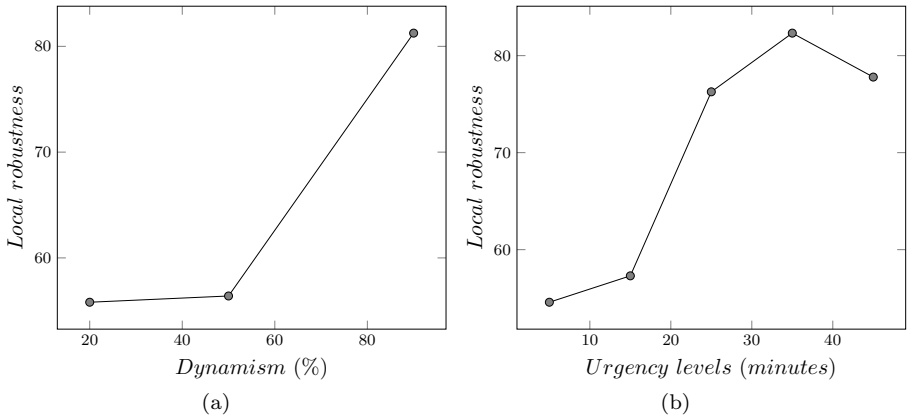


Figure 4: Local robustness of the periodic reoptimization policy.

Figure 4 shows how better local robustness is obtained for instances which are highly dynamic or which experience low urgency. The figure also illustrates the small decrease of 4% in the local robustness value when transitioning from urgency level 35 to 45. This may be attributed to the decreased flexibility of the scheduling algorithm during the very long reoptimization period when the large number of buffered requests demands more vehicles to achieve a higher robustness. Meanwhile, lower robustness levels are anticipated as the urgency increases further. The computational results fully support this prediction, as evidenced by the lack of a peak on the left-hand side of Figure 4(b). This effect is due to the existence of scenarios which incorporate queues of highly urgent requests that are larger than the fleet of available vehicles, which implies less flexibility and thus less robustness. Non-urgent scenarios with any dynamism degree appear the most flexible scenarios for the scheduling algorithms to address. This is possibly due to the balanced number of requests associated with such scenarios, a demand pattern which does not require a large fleet of vehicles. An average local robustness of 70% is achieved across all instances.

Global robustness Figure 5 provides the global robustness results and illustrates how global robustness for all instances remains in the range of 66-78%. Figure 5(b) shows that the global robustness decreases in a nonlinear fashion when the reoptimization frequency is increased. Nevertheless, global robustness across all instances is relatively high, averaging 73%.

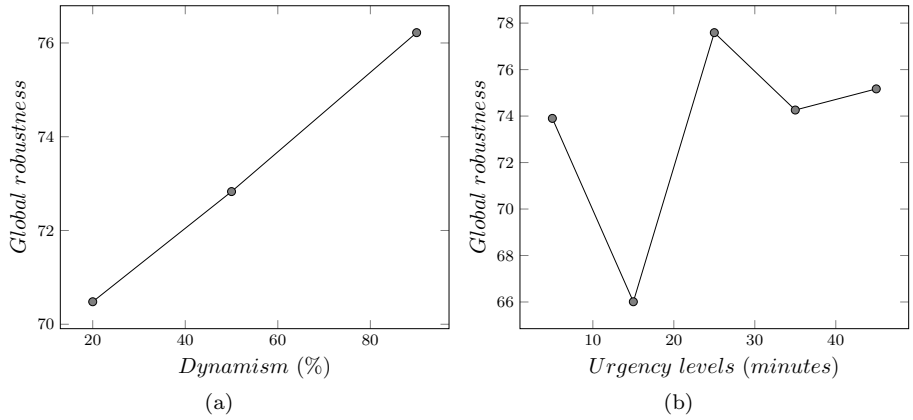


Figure 5: Global robustness of the periodic reoptimization policy.

4 Conclusions

This paper evaluated the robustness of a periodic reoptimization policy for the dynamic PDPTW with regard to varying reoptimization periods and dynamism degrees. The analysis indicates that robustness increases globally at the end of an instance's horizon or locally after each optimization when either the reoptimization period or the dynamism degree increases.

Shortening the reoptimization period may increase the frequency of disruptions, which incurs nervous decision-making without improving the schedule's quality. Another way of putting this is that a periodic reoptimization policy's response may be perfectly sufficient to deal with the disruptions faced and that rescheduling more frequently can prove counterproductive, at least from a robustness perspective. Therefore, the reoptimization frequency should be low enough to ensure a high degree of robustness. As soon as robustness begins to decrease the reoptimization frequency must be increased. If this simple rule is applied then the robustness of a periodic reoptimization policy will increase.

Acknowledgements

This research is funded by FWO as part of ORDinL (Operational Research and Data science in Logistics) project under the Strategic Basic Research (SBO) program. We also acknowledge the support provided by KU Leuven C24/17/012 "Dynamic Combinatorial Optimization". Editorial consultation provided by Luke Connolly (KU Leuven).

References

Gendreau, M., Guertin, F., Potvin, J.-Y., and Séguin, R. (2006). Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. *Transportation Research Part C: Emerging Technologies*, 14(3):157–174.

Karami, F., Vancroonenburg, W., and Vanden Berghe, G. (2018). A buffering-strategy-based solution method for the dynamic pickup and delivery problem with time windows. In *12th International Conference on the Practice and Theory of Automated Timetabling, Vienna, Austria, 28-31 August*, pages 256–272.

Karami, F., Vancroonenburg, W., and Vanden Berghe, G. (2019). Periodic vs. reactive scheduling for internal hospital logistics. *TR, KU Leuven*.

Morlok, E. K. and Chang, D. J. (2004). Measuring capacity flexibility of a transportation system. *Transportation Research Part A: Policy and Practice*, 38(6):405–420.

Psaraftis, H. N. (1988). Dynamic vehicle routing problems. *Vehicle routing: Methods and studies*, 16:223–248.

Srour, F. J., Agatz, N., and Oppen, J. (2016). Strategies for handling temporal uncertainty in pickup and delivery problems with time windows. *Transportation Science*, 52(1):3–19.

van Lon, R. R., Ferrante, E., Turgut, A. E., Wenseleers, T., Vanden Berghe, G., and Holvoet, T. (2016). Measures of dynamism and urgency in logistics. *European Journal of Operational Research*, 253(3):614–624.

Vancroonenburg, W., Esprit, E., Smet, P., and Vanden Berghe, G. (2016). 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*, pages 371–383.